WEAVER, BETTER, FASTER, LONGER: AN EFFECTIVE WORLD MODEL FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The potential impacts of world models (WMs, i.e., learned simulators) on robotics are far-reaching—policy evaluation, policy improvement, and test-time planning—all with limited real-world interaction. To unlock these downstream capabilities, a WM needs to jointly satisfy three desiderata: *(i)* fidelity (i.e., producing simulated trajectories that correlate with reality), *(ii)* consistency (i.e., producing simulated trajectories that are coherent over long horizons), and *(iii)* efficiency (i.e., producing simulated trajectories quickly). We propose WEAVER (World Estimation Across Views for Embodied Reasoning): a WM architecture that simultaneously achieves all three desiderata, providing state-of-the-art results on robotic manipulation tasks. WEAVER is a multi-view WM trained to predict future latents and reward values via a flow-matching loss. We distill the key design decisions across model architecture, memory, and prediction objectives required to unlock the kinds of long-horizon dynamic manipulation tasks that have confounded prior world modeling approaches. We apply WEAVER in robotic hardware, demonstrating its effectiveness at policy evaluation ($\rho = 0.817$ correlation with real-world success rate), policy improvement (real-world success rate improvement of 38% on top of the $\pi_{0.5}$ robot foundation model), and test-time planning (real-world success rate improvement of 14% made possible by upto $20\times$ inference speedup over prior WMs). WEAVER also demonstrates better performance than prior WMs when evaluated on out-of-distribution scenarios. Videos at https://weaver-world-model.github.io.

1 INTRODUCTION

World models (WMs, (Ha & Schmidhuber, 2018)), or learned simulators, have attracted intense interest from both academia (Guo et al., 2026b; Yin et al., 2026; Zhou et al., 2025) and industry (Bruce et al., 2024; Russell et al., 2025). Much of this interest stems from the promise of WMs for robotics: the ability to both evaluate and improve policies without costly and often unsafe real-world interaction. Furthermore, WMs unlock test-time scaling when incorporated into planning algorithms.

To simultaneously deliver on the three applications of evaluation, improvement, and planning, a robot WM must jointly satisfy the three core desiderata. The first is *(i) fidelity*: producing physically accurate predictions that correlate with real-world outcomes. The second is *(ii) consistency*: producing predictions that remain coherent over long prediction horizons. The third is *(iii) efficiency*: producing predictions quickly. For example, policy evaluation and improvement require high-fidelity predictions (for handling arbitrary, visuomotor robot policies) as well as consistency (for multi-stage tasks). Relatedly, planning requires fast inference for dealing with the real-time constraints of robots.

Despite rapid progress, no existing robot WM satisfies all three desiderata in tandem. For example, video generation models (Mei et al., 2026; Guo et al., 2026b; Agarwal et al., 2025; Quevedo et al., 2026) produce high fidelity generations at the cost of low efficiency. Similarly, since JEPA-style WMs (Assran et al., 2023; 2025) are trained without reconstruction, their latents may not map back to images, which visuomotor policies take as input. When we focus on robotic manipulation, the world modeling problem becomes even more complex, as we must handle multiple views of the scene, infer occluded objects from history, and ensure high-fidelity predicted world states to capture contact-rich object dynamics. Handling these complexities often comes at the cost of efficiency, with state-of-the-art WMs for manipulation operating at far slower speeds to satisfy real-time control, precluding their use in test-time planning and making policy improvement computationally challenging.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

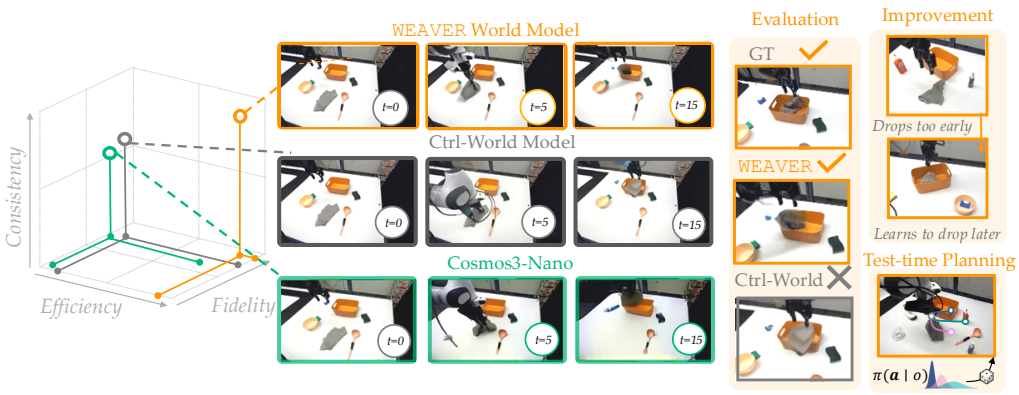


Figure 1: We present WEAVER, a WM that satisfies three desiderata: (i) high fidelity, (ii) long-horizon consistency and (iii) efficient generation. With these, we unlock the potential for downstream policy evaluation (middle), policy improvement (top right) and Test-time Planning (bottom right).

In response, we introduce WEAVER (**W**orld **E**stimation **A**cross **V**iews for **E**mbodied **R**easoning): a WM architecture that achieves *(i)* high fidelity, *(ii)* long-horizon consistency, and *(iii)* efficient generation. We show that WEAVER outperforms prior world models on generation quality and consistency, and that it can be effectively applied to policy evaluation, policy improvement, and test-time planning on challenging robotic manipulation tasks. To achieve these three capabilities, WEAVER combines flow matching (Lipman et al., 2023) with diffusion forcing (Chen et al., 2024), enabling long-horizon generation at fast inference speeds. WEAVER also uses a reward prediction head in the latent space of the world model (Hafner et al., 2021; Hansen et al., 2024; Jain et al., 2026) for evaluation without the need for reconstruction and an external judge model like a VLM (for efficient planning). Lastly, to handle the particular complexities of robot manipulation, WEAVER generates all camera views jointly and conditions on a memory of past observations (Guo et al., 2026b), keeping its predictions consistent across views and over time.

Together, these design choices yield a WM for robotic manipulation that Pareto-dominates methods such as Ctrl-World (Guo et al., 2026b) and Cosmos3-Nano (NVIDIA Cosmos Team, 2026) in the trade-off between generation quality and inference speed (Fig. 2). We show that WEAVER can be used flexibly across evaluation, improvement, and planning. On a suite of manipulation tasks (from pick and place to deformable object manipulation) performed on real hardware, WEAVER demonstrates strong correlation ($\rho = 0.817$) with real-world success rate when used for evaluation (Fig. 5), improves the real-world success rate of the $\pi_{0.5}$ (Intelligence et al., 2025) robot foundation model by 38% with WEAVER-generated data (Fig. 6), and unlocks test-time planning (Fig. 7). To our knowledge, WEAVER is the first WM framework to jointly demonstrate policy evaluation, offline policy improvement, and test-time planning with a pretrained VLA on real-world manipulation tasks.

2 Related Work

Robot World Models. While world models have been explored across autonomous driving (Russell et al., 2025; Wang et al., 2024), video games (He et al., 2025), and code generation (Copet et al., 2025), we focus on their application to robotics (Wu et al., 2023; Sharma et al., 2026; Team et al., 2025; Assran et al., 2025) – more specifically visual manipulation. While improvements in video generation (Wiedemer et al., 2025; Blattmann et al., 2023) have led to high *(i)* fidelity WMs (Mei et al., 2026; Guo et al., 2026b; Sharma et al., 2026; Quevedo et al., 2026; Guo et al., 2026b; Gao et al., 2026; Russell et al., 2025), these WMs are often not *(iii)* efficient enough to use for test-time planning. However, incorporating key ingredients from the broader vision community, like flow matching (Lipman et al., 2023), diffusion forcing (Chen et al., 2024) allows us to improve the *(iii)* efficiency of WEAVER. Furthermore, the use of pretrained video generation model encoders (Rombach et al., 2022) enhances WEAVER's robustness to out-of-distribution visual inputs, while the use of pretrained decoders allows us to evaluate arbitrary visuomotor robot policies unlike JEPA-style models (Assran et al., 2025). Finally, we adopt the latent reward and value heads of Dreamer-v4 (Hafner et al., 2025) to enable *(iii)* efficient evaluation and planning without the need to pass decoded images to an external and often slow VLM judge like in (Guo et al., 2026a).

Prior WMs (Hafner et al., 2021; Hansen et al., 2022; Wu et al., 2025; Jain et al., 2022) struggle to maintain temporal *(ii)* consistency across long horizons. In response, we adopt the use of multi-view prediction and memory from (Guo et al., 2026b; Russell et al., 2025) to ensure coherent generations even when gripper-object interactions are under occlusions. This is in contrast to WorldGym (Quevedo et al., 2026), DreamerV4 (Hafner et al., 2025) and DreamDojo (Gao et al., 2026).

Perhaps the most similar approaches to our own are Ctrl-World (Guo et al., 2026b) and Dreamer-v4 (Hafner et al., 2025). By using techniques from the video generation community (Chen et al., 2024; Lipman et al., 2023) for more *(iii)* efficient inference, we are able to produce higher *(i)* fidelity generations that are more temporally *(ii)* coherent in less time, Pareto dominating Ctrl-World. By using a pretrained encoder (Rombach et al., 2022) instead of learning from scratch as in Dreamer-v4 (Hafner et al., 2025), we likely inherit better robustness to out-of-distribution visual inputs.

Uses of World Models in Robotics. World models promise "downstream" advances in robotic policy evaluation, improvement, and test-time planning. Prior work has shown that sufficiently faithful world models can enable scalable policy evaluation (Team et al., 2025; Yin et al., 2026; Wang et al., 2026), while early results suggest that synthetic trajectories may also improve policies (Guo et al., 2026a; Wang et al., 2026), though the extent to which this is true remains an open question. More recently, world models have been explored for test-time planning (Qi et al., 2026; Wu et al., 2025), where the central challenge is generating accurately quickly for online optimization. `WEAVER` is designed with each of these downstream applications in mind for robotic manipulation.

3 World Estimation Across Views for Embodied Reasoning

We now describe the key ingredients in `WEAVER`: a robot world model designed to support policy evaluation, policy improvement, and test-time planning. These downstream applications of the WM on manipulation tasks imply three key desiderata upstream: *(i) fidelity* across multiple views during physical interaction, *(ii) consistent* predictions across long-horizon interactions that can introduce occlusions, and *(iii) efficient* enough generation for use in a real-time planning algorithm.

To jointly satisfy these three desiderata, `WEAVER` fuses together a variety of ingredients. We first describe the key WM design decisions and training objective (Sec. 3.1), followed by inference acceleration (Sec. 3.2) and latent-space value estimation (Sec. 3.3). We then show how, when put together, these components enable evaluation, improvement, and planning (Sec. 3.4).

Setup: Robot & Policy. We consider long-horizon robotic manipulation tasks specified by a natural language instruction $\ell \in \mathcal{L}$. Let the robot's proprioceptive state (e.g., joint angles) be denoted by $q \in \mathbb{R}^8$. The robot also has n RGB views of the scene (e.g., from wrist and third person cameras); let this set of multi-view images be $\mathbf{I} := (I^1, \ldots, I^n)$. At timestep t, the robot observes both the multiview images and proprioceptive state: $o_t := (\mathbf{I}_t, q_t) \in \mathcal{O}$. Let the robot's action be denoted by $a \in \mathcal{A}$ (e.g., joint velocities). Given any (o_t, ℓ), the robot's *base policy* π_θ generates $\mathbf{a}_t \sim \pi_\theta(\cdot \mid o_t, \ell)$, an h-step future action chunk (i.e., $\mathbf{a}_t := a_{t:t+h}$), which is then executed in the WM / environment.

World Model Architecture. Our WM maps an observation o_t into a latent state $z_t \in \mathcal{Z}$ via a pretrained encoder $z_t \sim \mathcal{E}_\psi(o_t)$. A key design choice is conditioning our world model on both a *memory* of every kth prior latent, $\mathbf{z}_t^{\texttt{mem}} := (\ldots, z_{t-2k}, z_{t-k})$, as well as a m-step *history* of the m most recent latents, $\mathbf{z}_t^{\texttt{hist}} := (z_{t-m}, \ldots, z_t)$. Given memory, history, and an h-step action plan $\mathbf{a}_t$, the WM predicts h *future latents*:

$$\hat{\mathbf{z}}_t \sim f_\phi(\cdot \mid \mathbf{z}_t^{\texttt{mem}}, \mathbf{z}_t^{\texttt{hist}}, \mathbf{a}_t), \tag{1}$$

where $\hat{\mathbf{z}}_t := \hat{z}_{t+1:t+h+1}$ is the h-step future. We also train a reward model that scores the predicted latent's alignment with the language instruction: $\hat{\mathbf{r}}_t \sim R(\cdot \mid \hat{\mathbf{z}}_t, \ell)$ where $\hat{\mathbf{r}}_t := \hat{r}_{t+1:t+h+1}$. To enable iterative calls to the visuomotor policy, we use a pretrained decoder to obtain future observations (camera views and proprioceptive state), $\hat{\mathbf{o}}_t \sim \mathcal{D}_\eta(\hat{\mathbf{z}}_t)$, where $\hat{\mathbf{o}}_t := \hat{o}_{t+1:t+h+1}$ are the h-step future observations. The final prediction, $\hat{o}_{t+h+1}$, is fed back to the policy to generate the next action chunk.

3.1 Key Design Decisions for High Fidelity, Temporally Consistent Generation

Multi-View Camera Prediction. Although using multiple views (e.g., wrist and external cameras) is common practice when designing visuomotor robot policies for manipulation to handle partial

observability and capture finer-grained object–gripper interactions (Intelligence et al., 2025), many WMs only predict a single view (Quevedo et al., 2026; Gao et al., 2026). Following (Guo et al., 2026b; Jiang et al., 2025; Wang et al., 2025), WEAVER predicts both external and wrist-camera observations. The increased information provided by multiple views helps with *(ii)* consistency by helping handle occlusions during manipulation. Each view I_t^j is encoded into $H \times W$ patch tokens using the pretrained SD3 encoder (Esser et al., 2024). We project the proprioceptive state q_t to the same token dimension and obtain z_t by concatenating patch tokens and the proprioceptive token.

Proprioceptive State Prediction. In addition to future visual latents, WEAVER also predicts future proprioceptive states. We find that explicitly predicting the robot's configuration (rather than just visual observations like Ctrl-World) is helpful to handle contact-rich manipulation of deformable objects, where knowing the precise position of the arm and width of the gripper is often required.

Sparse Memory and Short-Term History. Temporal *(ii)* consistency across WM generations requires the WM to understand both what changes and what stays the same across an interaction. This is particularly challenging in manipulation, where occlusions and wrist camera viewpoint changes can cause objects and parts of the background scene to leave and enter the robot's FOV. In response, WEAVER builds upon Ctrl-World and conditions on two sets of observations: a long-term, sparse *memory*, and a short-term *history*. In particular, memory $\mathbf{z}_t^{\texttt{mem}} := (\dots, z_{t-2k}, z_{t-k})$ includes every kth observation to capture longer-term context, while history $\mathbf{z}_t^{\texttt{hist}} := (z_{t-1}, z_t)$ includes the last two frames to capture the shorter-term consequences of actions.

Latent Dynamics Model. The latent dynamics model $\hat{\mathbf{z}}_t \sim f_\phi(\cdot \mid \mathbf{z}_t^{\texttt{mem}}, \mathbf{z}_t^{\texttt{hist}}, \mathbf{a}_t)$ predicts future latent states conditioned on memory, history, and a candidate action plan. To balance *(i)* fidelity with *(iii)* efficiency, WEAVER adopts an efficient 2D transformer architecture following (Hafner et al., 2025; Quevedo et al., 2026), with L dynamics blocks composed of spatial attention and causal temporal attention. At each prediction step, the model conditions on latent tokens, action tokens, and flow timestep embeddings to autoregressively generate an h-step chunk. For stable training, each block uses RMSNorm (Zhang & Sennrich, 2019), RoPE (Su et al., 2024), QKNorm (Henry et al., 2020), and SwiGLU feed-forward layers (Shazeer, 2020) (see A2 for more details).

Training Objective. Similar to (Hafner et al., 2025), we train our latent dynamics model f_ϕ with a flow-matching loss (Lipman et al., 2023) to predict future latents. Let $x_t^1 := z_{t+1:t+h+1}$ denote the ground-truth next h latents and let $x_t^0 \sim \mathcal{N}(0, I)$ denote a Gaussian noise vector of the same dimension. Next, we define $x_t^\tau = \tau x_t^1 + (1-\tau)x_t^0$, with $\tau \in [0, 1)$. Then, we train f_ϕ to predict "velocity" $x_t^1 - x_t^0$ by minimizing mean squared error: $\mathcal{L}^{\texttt{WM}}(\phi) = \mathbb{E}_{x_t^0, x_t^1, \tau}\left[\left\|(x_t^1 - x_t^0) - f_\phi(\mathbf{z}_t^{\texttt{hist}}, \mathbf{z}_t^{\texttt{mem}}, \mathbf{a}_t, x_t^\tau, \tau)\right\|_2^2\right]$. To improve long-horizon *(ii)* consistency, we adopt Diffusion Forcing (Chen et al., 2024), which trains the latent dynamics model with independently sampled noise levels across future timesteps. We also use SPRINT blocks (Park et al., 2026), which aggressively drop patch tokens in the latents to improve *(iii)* efficiency.

3.2 Accelerating World Model Inference Speed

For diffusion transformer-based WMs (Guo et al., 2026b; Hafner et al., 2025; Quevedo et al., 2026) like WEAVER, latency is a product of both *(a)* the forward pass through the model and *(b)* iterative denoising. Thus, *(iii)* efficient generation requires tackling both of these concerns in tandem. We reduce cost *(a)* via the use of KV caching to memory and history tokens across denoising steps. We reduce cost *(b)* by adjusting the denoising process. In particular, building on diffusion forcing (Chen et al., 2024), we use a progressive noise schedule. Rather than using a linear schedule like in (Guo et al., 2026b; Hafner et al., 2025), WEAVER adopts a cosine schedule for higher *(i)* fidelity.

To further increase *(iii)* efficiency to the level required for test-time planning, we post-train WEAVER with a rectified flow objective (Liu et al., 2023) to enable high-quality generation within a few forward passes. In particular, we first generate a high-quality latent trajectory using the denoising process, before using it as a target for secondary distillation step. See Appendix A2.3 for more details.

3.3 Accurate and Efficient Value Estimation from the World Model

Reward Model. To enable *(iii)* efficient scoring of a proposed action chunk without needing to *(a)* decode a latent into an image and *(b)* feed it to an external VLM judge model (Guo et al.,

2026a; Quevedo et al., 2026), we distill the scores produced by an off-the-shelf reward model into a lightweight reward head that operates directly on latent states and language instruction ℓ. The reward head R aggregates latent tokens with AdaPool (Brothers, 2025), followed by MLP layers.

Critic. To support truncated-horizon rollouts with the WM, WEAVER learns a critic network V that estimates the value beyond the imagined horizon. The critic shares the same latent-space design as the reward model and is trained with an MSE objective to predict bootstrapped λ-returns (Sutton et al., 1998). Given latent rewards from R, the target is defined recursively as $\mathbf{v}_t^\lambda = R(z_t, \ell) + \gamma\Big((1-\lambda)V(z_{t+1}, \ell) + \lambda \mathbf{v}_{t+1}^\lambda\Big)$, $\mathbf{v}_{t+k}^\lambda = V(z_{t+k}, \ell)$. The critic is then trained by minimizing $\mathcal{L}^{\text{critic}}(V) = \left\| V(z_t, \ell) - \mathbf{v}_t^\lambda \right\|_2^2$.

3.4 Downstream WM Applications: Evaluation, Improvement, Planning

By satisfying the desiderata of *(i)* fidelity, *(ii)* consistency, and *(iii)* efficiency simultaneously, WEAVER can support the downstream capabilities of evaluation, improvement, and planning.

Policy Evaluation. For policy evaluation, we take recorded action trajectories from real-world rollouts and execute them open-loop inside WEAVER, recording human preference as success labels. We focus on long-horizon tasks that sometimes require 40+ iterative evaluations of WEAVER's latent dynamics model, underscoring the importance of temporal *(ii)* consistency and *(iii)* efficiency.

Policy Improvement. We sample a h-step action chunk from the policy and forward simulate inside the WM K times for a total of $H = Kh$ timesteps, leveraging WEAVER's *(i)* fidelity and *(ii)* consistency. After doing this B times from the same initial observation z_t, we collect batch of rollouts $\{(z_t, a^b_{t:t+H-1}, \hat{z}^b_{t+1:t+H})\}_{b=1}^B$. We then compute a Monte-Carlo estimate of the H-step *advantage* along each rollout: $\hat{A}_t^b = \sum_{j=1}^H \gamma^{j-1} R(\hat{z}^b_{t+j}, \ell) + \gamma^H V(\hat{z}^b_{t+H}, \ell) - V(z_t, \ell)$. If the highest-scoring rollout in the batch (i.e., $b^\star = \arg\max_{b\in\{1,\dots,B\}} \hat{A}_t^b$) has an advantage value above some small, positive threshold (i.e., $\hat{A}_t^{b^\star} > \epsilon_{\text{adv}}$), we distill it into the base policy. This advantage-based filtering prevents the policy from being updated at states where all H-step sampled plans are predicted to be worse than the current expected behavior of the policy (Jain et al., 2026; Anthony et al., 2017).

Test-time Planning. We adopt a single-chunk, best-of-N (Lightman et al., 2024) approach to test-time scaling that doesn't involve iteratively calling the latent dynamics model. In particular, given the current observation and instruction, we sample B candidate action chunks from the policy, imagine their outcomes with the world model, and execute the one with the highest advantage estimated with latent reward and critic heads. WEAVER's *(iii)* efficiency (both in terms of the speed of the latent dynamics model and ability to evaluate a candidate action sequence without needing an external VLM judge via the use of the reward head) are critical to unlocking this test-time scaling capability.

4 Experimental Setup

Base Policy & Hardware. Our base policy is $\pi_{0.5}$ (Intelligence et al., 2025), a state-of-the-art vision-language-action (VLA) policy trained on DROID dataset (Team, 2024). We follow the DROID hardware setup and use a single Franka Emika Panda manipulator, two external Zed 2i cameras mounted on the left and right sides of the workspace, and a wrist-mounted Zed Mini camera (Fig. 8). The $\pi_{0.5}$ policy and our WEAVER world model use only the right camera view and the wrist camera[1].

Datasets & Tasks. To align the world model with the data distribution of the base policy, we first pretrain the WEAVER world model on the DROID dataset and then finetune it on our real-world setup. We collect data to fine-tune the world model $\mathcal{D}^{\text{FT}}_{\text{real}}$ by running $\pi_{0.5}$ for five real-world manipulation tasks, with 50 rollouts per task. We also collect an additional 20 rollouts per task as evaluation data $\mathcal{D}^{\text{val}}_{\text{real}}$. We select tasks such that the base policy achieves at least 20% success rate while spanning a range of capabilities from rigid object pick-and-place to deformable object manipulation and dynamic manipulation. Specifically, our tasks are: ***Stack Bowls*** (stack one bowl on another); ***PnP Bag*** (place a deformable chip bag onto a plate); ***PnP Marker*** (reorient a marker and insert it into a cup); ***PnP Towel*** (place a soft towel into a basket); and ***Pour Beans*** (pour a cup full of coffee beans into a bowl). Details on each task can be found in Appendix A1.1.

[1]We setup all three cameras because our main world model baseline (Guo et al., 2026b) uses all three views.

	DROID (val)				*Task data (OOD)*				**Time**
	Exterior		*Wrist*		*Exterior*		*Wrist*		
Method	**FID** ↓	**FVD** ↓	**FID** ↓	**FVD** ↓	**FID** ↓	**FVD** ↓	**FID** ↓	**FVD** ↓	*(s)* ↓
Cosmos3-Nano	34.13	137.20	32.11	190.07	42.44	182.39	42.23	295.09	16.30
Ctrl-World	26.09	78.73	33.83	195.37	36.16	139.54	38.76	277.13	14.65
WEAVER	**10.20**	**27.83**	**21.50**	**90.72**	**23.95**	**88.27**	**30.77**	**184.62**	**4.78**

Table 1: We report FID and FVD on DROID(val) and OOD datasets at NFE=16. We observe that WEAVER outperforms baselines on fidelity and has lower inference time.

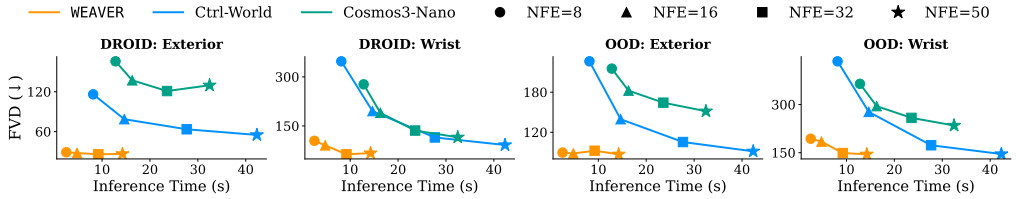


Figure 2: We present FVD vs inference time (seconds) across views on DROID and OOD datasets, where WEAVER pareto-dominates baselines for different NFEs while using 16× less inference time.

World Model Training. WEAVER is a 928M parameter model and is pretrained on the DROID for 1M steps with a batch size of 32 and learning rate of $1e^{-4}$ on $4 \times H100$ GPUs for 10 days. For training the reward model and critic on top of WEAVER's latents, we annotate the DROID dataset with progress-rewards obtained from Robometer Liang et al. (2026) (reduced by 1 to get negative rewards). During world model finetuning, the model is updated with a lower learning rate of $2e^{-5}$ for 16k steps on our collected task data. The resulting model is used for policy evaluation, policy finetuning, and test-time planning. We represent actions as the joint position difference between two timesteps to match the action space of the $\pi_{0.5}$ policy. We learn an additional joint-velocity-to-position action adapter to convert between the action spaces for data generation and test-time planning (see A1.2).

5 Results

We first study the performance of the WEAVER world model in isolation (Sec. 5.1) and then in the downstream use-cases of policy evaluation, improvement, and test-time planning (Sec. 5.2).

5.1 WEAVER Pareto-Dominates Leading Manipulation World Models

We start by comparing the performance of WEAVER pretrained only on the DROID dataset to leading multi-view manipulation world models. Ctrl-World (Guo et al., 2026b) is a 1.5B-parameter model trained on the DROID dataset and initialized from a pretrained SVD model (Blattmann et al., 2023). We also compare against Cosmos3-Nano (NVIDIA Cosmos Team, 2026), a 14B-parameter multimodal video foundation model trained on diverse video and embodied-data sources.

Setup & Metrics. We evaluate on a validation split of the DROID dataset (256 trajectories) and an out-of-distribution dataset $\mathcal{D}_{\text{real}}^{\text{val}}$ collected using $\pi_{0.5}$ VLA (100 trajectories). For each trajectory, the models are rolled out autoregressively to generate 10s long sequences where each generation predicts the outcome of 15-step action chunks (1s) jointly. Following prior evaluations (Guo et al., 2026b), we measure the visual fidelity of the decoded generations using FID (Heusel et al., 2017), and FVD (Unterthiner et al., 2018) computed with the ground-truth videos (more details in A3.1).

Results: Perceptually High-Fidelity Generations. Table 1 compares WEAVER and baselines across evaluation datasets. WEAVER outperforms Ctrl-World and Cosmos3-Nano while requiring less inference time. In Table A3.2, we present that reducing number of function evaluations (NFE) degrades the baselines substantially more than WEAVER. We provide additional results on noise schedules (A3.4), inference speedups from KV caching (A3.3), and visualizations of action controllability (A3.1).

Results: WEAVER Pareto-Dominates Inference Speed vs. Generation Quality. Next, we study how generation quality is influenced by a fixed inference-time budget, measured by NFEs and the

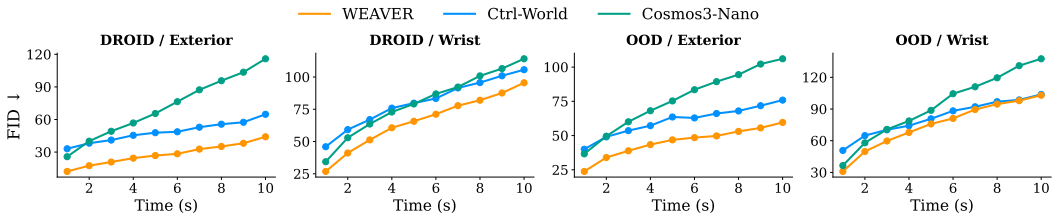


Figure 3: We report FID at various horizon lengths (in seconds) and observe that WEAVER is consistently better at long-horizon rollouts compared to Ctrl-World and Cosmos3-Nano.

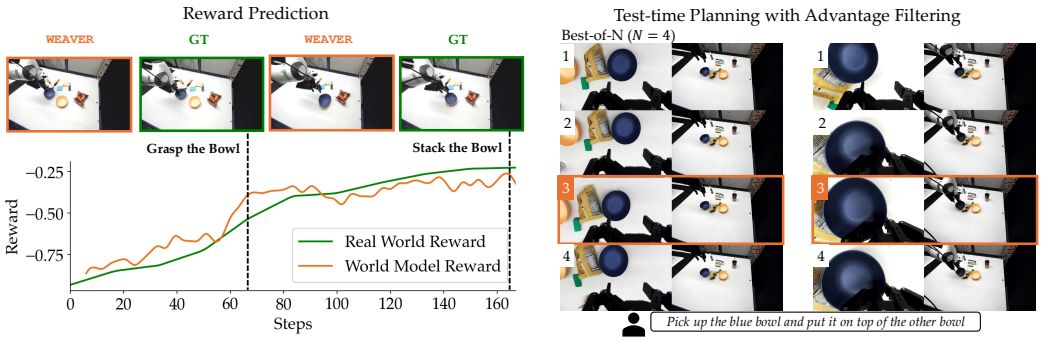


Figure 4: (Left) Predicted rewards from WEAVER match the Robometer reward. (Right) The highlighted sample is the one with the best advantage value and the best outcome in WM's imagination.

inference time (in seconds) required to generate a 10s chunk on a single H100 GPU. As shown in Fig. 11, WEAVER consistently outperforms Ctrl-World and Cosmos3-Nano across NFEs of 8,16,32,50 while requiring up to 5× less inference time at the same NFE. Notably, WEAVER at NFE=16 achieves comparable or better performance than both Ctrl-World and Cosmos3-Nano at NFE=50, yielding a 9× inference speedup. Interestingly, WEAVER surpasses Cosmos3-Nano across inference budgets, despite the latter having 14B parameters and being pretrained on a larger dataset. By pareto-dominating the baselines, WEAVER unlocks faster evaluation and planning as we explore below in Sec. 3.4.

Results: Higher Quality at Long Horizon. We next measure how the perceptual quality of the world model's imaginations are influenced by long-horizon predictions. We generate rollouts with long (150-step or 10s) action sequences and measure performance for predicted videos at regular interval to estimate the generation quality with time horizon. On the DROID dataset, we find that WEAVER maintains consistently lower metrics compared to baselines (Fig. 3 & 11). On the OOD dataset, WEAVER maintains the performance gap on exterior-view and has comparable performance to Ctrl-World on the wrist-view. Lastly, Cosmos3-Nano degrades more substantially with prediction horizon, likely because it lacks an explicit memory mechanism and generates frames autoregressively.

Results: Latent Reward Prediction Accuracy. We compare WEAVER's latent reward prediction to the reward labels from RoboMeter (Liang et al., 2026). Fig. 4 shows the predicted reward for a rollout of the **Stack Bowl** task; WEAVER correctly imagines key events such as grasping and stacking and the reward of the imaginations correlates with the ground-truth RoboMeter reward. In Fig. 4 (right), we see that the advantage computed with the predicted reward is also able to distinguish different outcomes of the action samples. This is a promising indicator that WEAVER and its latent reward are suitable for filtering synthetic data in Sec. 5.2.1 and test-time planning in Sec. 5.2.3.

Results: BridgeV2 dataset. Finally, we provide results on the BridgeV2 dataset (Walke et al., 2023). In Table 8, we see that WEAVER outperforms leading methods, including a 3.6× reduction in FVD over WorldGym (Quevedo et al., 2026) and a 10.7× reduction over IRASim (Zhu et al., 2025).

5.2 WEAVER Enables Effective Evaluation, Improvement and Planning

Thus far, we have validated that WEAVER effectively balances *(i)* fidelity, *(ii)* long-horizon consistency, and *(iii)* efficient generation. Next, we turn to the downstream uses of a world model: policy evaluation (Sec. 5.2.1), policy improvement (Sec. 5.2.2), and test-time planning (Sec. 5.2.3).

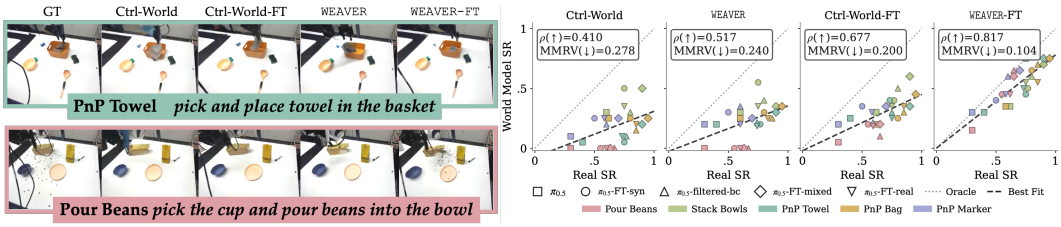


Figure 5: (Left) For PnP Towel, only WEAVER and WEAVER−FT accurately imagine the towel inside the basket. For Pour Beans, only WEAVER−FT captures the beans scattering across the table. (Right) Evaluation inside WEAVER-FT attains high rank correlation with real-world success rates.

5.2.1 Effective Policy Evaluation that Tightly Correlates with Reality

First, we evaluate whether WEAVER can serve as a learned simulator for offline policy evaluation, reducing the need for costly real-world rollouts.

Setup. Given an initial real observation, o_0, and action sequence, $\mathbf{a}_t \sim \pi_\theta(\cdot \mid o_0, \ell)$, we autoregressively generate imagined observations and estimate policy performance from the resulting rollout. We compare four world models: Ctrl-World pretrained on DROID (Guo et al., 2026b), WEAVER pretrained on DROID, Ctrl-World-FT and WEAVER−FT finetuned on $\mathcal{D}_{\text{real}}^{\text{FT}}$. To test robustness of the world models across base policy quality, we evaluate each model on rollouts from five different policies including the base $\pi_{0.5}$ policy $\pi_{0.5}$, finetuned policies **FT w/ Real Data**, **FT w/ Synthetic Data**, **FT w/ Mixed Data** and **FT w/ Filtered BC**. Details of policies are in Sec. 5.2.2 and App. A5.1.

Metrics. Following Yin et al. (2026), we measure how well performance of generations correlates with real-world performance by comparing human-labeled binary success rates on imagined rollouts with real success rates on $\mathcal{D}_{\text{real}}^{\text{val}}$ (20 trials per task). We report Pearson Correlation coefficient (Pearson, 1920) and maximum matrix ranking violation (MMRV) (Team et al., 2025) (see A5.1).

Results. Fig. 5 shows that pretrained world models tend to underestimate policy performance, but WEAVER achieves better agreement with real rollouts than Ctrl-World, with higher rank correlation (ρ = 0.517 vs. 0.410) and lower MMRV (0.240 vs. 0.278). This is challenging because rollouts can last up to 40 seconds and require accurate long-horizon prediction. The pouring task is particularly difficult for pretrained models, likely because granular dynamics are underrepresented in DROID and inherently hard to model. After finetuning, WEAVER-FT greatly improves evaluation accuracy, increasing the Spearman correlation to ρ = 0.817 (Pearson r = 0.846) and better matching real outcomes across policies of varied performance. The qualitative example in Fig. 5 (left) further shows WEAVER−FT predicts the **PnP Towel** and **Pour Beans** task outcomes more accurately than baselines.

5.2.2 WEAVER Enables Effective Policy Improvement Without Real Interactions

Another desirable use of high-fidelity world models is synthetic data generation for policy improvement. We use the world model to sample and verify candidate action segments, then distill high-value imagined segments back into the policy Jain et al. (2026); Anthony et al. (2017).

Setup. To evaluate the utility of WEAVER towards improving policies, we explore various strategies to generate data for finetuning the policy: (1) **Base Policy**: the original $\pi_{0.5}$ VLA trained on DROID; (2) **FT w/ Real Data**: we prune segments in real trajectories using advantage estimates, yielding 1,000 segments of 36-step action chunks per task; (3) **FT w/ Synthetic Data**: we sample multiple segments using the base policy and WEAVER, filter them based on predicted advantage values (Sec. 3.4), and retain 1,000 segments per task, and (4) **FT w/ Mixed Data**: combine filtered real and synthetic datasets (2000 segments per task) (more details and results are presented in App. A2.2 & A5.3).

Results. Fig. 6 shows that all finetuned policies substantially improve their success rate over the base policy. Notably, finetuning on synthetic data closely matches that on real data, with only a 4% average performance gap. This indicates that our synthetic data is of such a high quality that it unlocks similar policy improvement to costly real world data. Combining real and synthetic data further improves performance, increasing the average success rate by 11% over real-data finetuning alone. These results suggest that imagined rollouts from the world model provide a useful source for distillation, reducing the need for costly real-world collection and manual filtering. Fig. 6 also shows

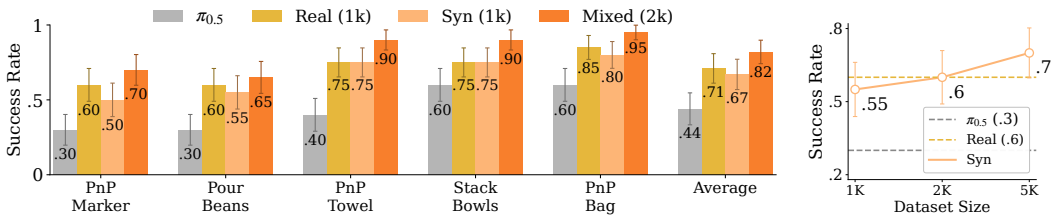


Figure 6: (Left) We finetune $\pi_{0.5}$ with multiple data sources and combining real and synthetic (Syn) obtained with WEAVER outperforms other variants. (Right) We ablate the number of segments in synthetic data for finetuning and report the success rate across 20 trials for the **Pour Beans** Task.

improvements on contact-rich and dynamic manipulation tasks, such as marker placement and bean pouring. We further study synthetic data scaling on the **Pour Beans** task by varying the number of imagined segments from 1K to 5K. Fig. 6 (right) shows that policy improves consistently with more synthetic data, eventually *exceeding the performance obtained from real-data finetuning alone*.

5.2.3 WEAVER ENABLES TEST-TIME PLANNING BY BALANCING LATENCY AND QUALITY

Finally, test-time search requires evaluating multiple action sequences before execution, making inference speed a key bottleneck. In contrast to planning in the image space using reconstruction with VLM-as-a-judge (Guo et al., 2026a; Sharma et al., 2026), WEAVER plans in the latent space for greater efficiency Hafner et al. (2021); Jain et al. (2026).

Setup. We use $\pi_{0.5}$ as the base policy and sample a batch of action chunks. For each chunk, WEAVER imagines latents of future states and evaluates the advantage using the reward and critic heads. This reduces the cost of decoding predicted observations and querying external VLM judges. Following the policy-improvement setup from Sec. 5.2.2, we evaluate test-time planning on five tasks and compare against the base policy. We use $B = 4$ parallel samples and horizon of $h = 12$, balancing planning quality and latency.

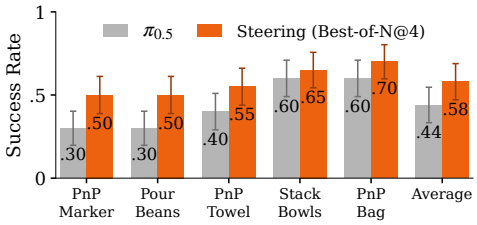


Figure 7: We show test-time steering with WEAVER outperforms the $\pi_{0.5}$ by 14%.

Results. We report task success rate and the inference-time breakdown in the test-time planning pipeline. Fig. 7 shows that advantage-based selection steers the policy toward successful behaviors. Test-time planning improves the average success rate by 15% over the base policy, with maximum gain up to 20%. The improvement is larger when the base policy is weaker, although it remains smaller than direct finetuning because planning horizon is limited to a single action chunk and must operate under latency constraints. More inference time analysis from Table. 7 in Appendix. A5.4 shows Reflow (NFE=4) makes WEAVER about 20× faster than Ctrl-World inference with their default parameters while obtaining higher quality. On an RTX A6000 Ada GPU, with batch size of 1, we achieve 33.5 FPS for future generation and we also obtain 10.3 Hz control frequency for test-time planning with batch size of 4. Batched sampling scales sublinearly with the number of candidates, showing our inference optimizations make model-based test-time planning practical for manipulation.

6 CONCLUSION

We introduce WEAVER: a World Model for manipulation that achieves *(i)* high fidelity, *(ii)* is temporally coherent, and *(iii)* generates efficiently. While WEAVER unlocks the potential of large-scale world models for manipulation, several limitations remain. First, visual world models observe only a partial view of the underlying state, and tactile sensing may be necessary to resolve ambiguities. Second, incorporating physics priors could improve performance on tasks involving deformable-object manipulation. Third, generation latency currently limits test-time planning to short-horizon reasoning over a single action chunk. Finally, reward supervision from RoboMeter can be noisy, motivating the development of better reward models for failure prediction. We provide further discussion in A6.

AI USE STATEMENT

We used generative AI tools to assist with implementing baselines and evaluation metrics for this work. The initial implementation of the WEAVER architecture and its downstream applications was written by the authors. The generative AI tools were subsequently used to assist with debugging and code editing. All AI-assisted code changes were manually reviewed and tested by the authors. We did not use generative AI tools to formulate hypotheses, design experiments, develop the methodology, formulate mathematical claims, or interpret experimental results. We additionally used generative AI tools to identify grammatical errors and improve the readability of the manuscript.

REPRODUCIBILITY STATEMENT

Appendix A2 presents the architectural details and hyperparameters required to train WEAVER. We also provide details on the training, fine-tuning, and post-training of the world model (WM). In Appendix A1.1, we describe the real-robot setup and the tasks used in this work, along with details of the action adapter module. For all baselines, we use the official implementations and checkpoints. We plan to release our code upon publication.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *CoRR*, abs/2506.09985, 2025.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *CoRR*, abs/2311.15127, 2023.

Greyson Brothers. Robust noise attenuation via adaptive pooling of transformer outputs. In *International Conference on Learning Representations (ICLR)*, 2025.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Neural Information Processing Systems (NeurIPS)*, 2024.

Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, et al. Cwm: An open-weights llm for research on code generation with world models. *CoRR*, abs/2510.02387, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning (ICML)*, 2024.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *CoRR*, abs/2602.06949, 2026.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *CoRR*, abs/2602.12063, 2026a.

Yanjiang Guo, Lucy Xiaoyang Shi, Jianyu Chen, and Chelsea Finn. Ctrl-world: A controllable generative world model for robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2026b.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations (ICLR)*, 2021.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *CoRR*, abs/2509.24527, 2025.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, et al. Matrix-game 2.0: An open-source real-time and streaming interactive world model. *CoRR*, abs/2508.13009, 2025.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *CoRR*, abs/2504.16054, 2025.

Arnav Kumar Jain, Shiva Kanth Sujit, Shruti Joshi, Vincent Michalski, Danijar Hafner, and Samira Ebrahimi Kahou. Learning robust dynamics through variational sparse gating. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled SAILOR: Robust imitation via learning to search. In *Neural Information Processing Systems (NeurIPS)*, 2026.

Yuxin Jiang, Shengcong Chen, Siyuan Huang, Liliang Chen, Pengfei Zhou, Yue Liao, Xindong He, Chiming Liu, Hongsheng Li, Maoqing Yao, et al. Enerverse-ac: Envisioning embodied environments with action condition. *CoRR*, abs/2505.09723, 2025.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *Robotics: Science and Systems 2026*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhiting Mei, Tenny Yin, Ola Shorinwa, Apurva Badithela, Zhonghe Zheng, Joseph Bruno, Madison Bland, Lihan Zha, Asher Hancock, Jaime Fernández Fisac, et al. Video generation models in robotics-applications, research challenges, future directions. *CoRR*, abs/2601.07823, 2026.

NVIDIA Cosmos Team. Cosmos 3: Omnimodal world models for physical ai. Technical report, 2026. URL https://research.nvidia.com/labs/cosmos-lab/cosmos3/technical-report.pdf.

Dogyun Park, Moayed Haji-Ali, Yanyu Li, Willi Menapace, Sergey Tulyakov, Hyunwoo J Kim, Aliaksandr Siarohin, and Anil Kag. Sprint: Sparse-dense residual fusion for efficient diffusion transformers. In *International Conference on Learning Representations (ICLR)*, 2026.

Karl Pearson. Notes on the history of correlation. *Biometrika*, 13(1):25–45, 1920.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 11(5):5534–5541, 2026.

Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation. In *International Conference on Learning Representations (ICLR)*, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving (2025). *CoRR*, abs/2503.20523, 2025.

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. Worldgymnast: Training robots with reinforcement learning in a world model. *CoRR*, abs/2602.02454, 2026.

Noam Shazeer. Glu variants improve transformer. *CoRR*, abs/2002.05202, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

DROID Team. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

Gemini Robotics Team, Krzysztof Choromanski, Coline Devin, Yilun Du, Debidatta Dwibedi, Ruiqi Gao, Abhishek Jindal, Thomas Kipf, Sean Kirmani, Isabel Leal, et al. Evaluating gemini robotics policies in a veo world simulator. *CoRR*, abs/2512.10675, 2025.

Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *CoRR*, abs/1812.01717, 2018.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-drive world models for autonomous driving. In *European Conference on Computer Vision (ECCV)*, 2024.

Yixuan Wang, Rhythm Syed, Fangyu Wu, Mengchao Zhang, Aykut Onol, Jose Barreiros, Hooshang Nayyeri, Tony Dear, Huan Zhang, and Yunzhu Li. Interactive world simulator for robot policy training and evaluation. *CoRR*, abs/2603.08546, 2026.

Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *CoRR*, abs/2509.20328, 2025.

Clark Wissler. The spearman correlation formula. *Science*, 22(558):309–311, 1905.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning (CoRL)*, 2023.

Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. In *Robotics: Science and Systems (RSS)*, 2025.

Tenny Yin, Zhiting Mei, Zhonghe Zheng, Miyu Yamane, David Wang, Jade Sceats, Samuel M Bateman, Lihan Zha, Apurva Badithela, Ola Shorinwa, et al. Playworld: Learning robot world models from autonomous play. *CoRR*, abs/2603.09030, 2026.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning (ICML)*, 2025.

Fangqi Zhu, Hongtao Wu, Song Guo, Yuxiao Liu, Chilam Cheang, and Tao Kong. Irasim: A fine-grained world model for robot manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9834–9844. IEEE, 2025.

A1 Robot Setup & Tasks

A1.1 Tasks Details

We collect real-world finetuning data from $\pi_{0.5}$ on five manipulation tasks, with 50 rollouts per task on our DROID setup as shown in Fig. 8. We select tasks for which the base policy achieves at least 20% success, ensuring that the collected rollouts contain both successful and failed executions while remaining within the policy's competence. The tasks are designed to cover a diverse set of manipulation regimes, including rigid-object pick-and-place, deformable-object manipulation, and dynamic manipulation as shown in Fig. 8.

Stack Bowls requires the robot to stack one bowl on top of another. Two bowls are randomly placed on the table, and the robot must place bowl A on bowl B, where $A \in \{\text{blue}, \text{green}\}$ and $B \in \{\text{blue}, \text{green}, \text{pink}\} \setminus A$.

PnP Bag requires the robot to pick up a bag of chips and place it on a green plate. We use two types of chip bags and randomly sample one in each episode. The bag is deformable, making the grasp outcome and object motion difficult to predict.

PnP Marker requires the robot to pick up an Expo marker lying horizontally on the table and place it inside a container. The marker color is randomly selected from black and orange, and the target container is randomly selected from a paper cup and a blue mug. This task requires precise grasping and large end-effector reorientation to insert the marker vertically.

PnP Towel requires the robot to pick up a towel and place it into a basket. We use two towel variants, a folded thick red kitchen towel and a thin gray square towel, and two basket variants, orange and blue. The task is challenging because the towel is deformable, and its resulting shape depends strongly on the grasp location and, for the folded towel, the number of layers grasped.

Pour Beans requires the robot to pick up a cup containing coffee beans and pour them into a blue bowl. This task tests dynamic manipulation, as the granular motion of the beans is difficult to predict and successful execution requires accurate control of cup pose, pouring angle, and motion to avoid spilling outside the bowl.

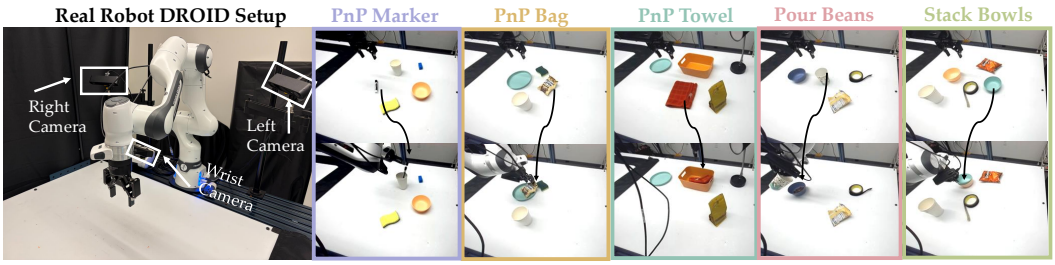


Figure 8: **Hardware setup and tasks.**On the left, it is the robot setup with cameras. On the right, it shows the five tasks with top row as initial state and bottom row as one of the goal configuration.

A1.2 Action Space

The $\pi_{0.5}$ base policy on the DROID setup outputs joint-velocity commands for control. To match this action representation, we define the action space of our world model in joint space, avoiding potential compounding errors from converting actions into alternative representations such as Cartesian space. However, we find that directly conditioning the world model on joint velocities leads to lower generation quality. Therefore, following prior work Guo et al. (2026b), we use a lightweight action adapter to convert joint velocities into joint positions.

For policy evaluation, because the joint positions at the end of each trajectory are already available, we directly use joint positions as inputs to the world model. During test-time planning, however, predictions must be made from the joint-velocity actions proposed by the policy. We therefore use the trained action adapter to predict the corresponding joint positions, and condition the world model generation on these adapted actions. The following section describes the details of the action adapter.

A1.2.1 ACTION ADAPTER

Overview. The action adapter is a lightweight feedforward module that bridges the world model's action representation (joint velocity commands and binary gripper signals) and the robot's observable state (absolute joint positions and gripper width). Given the robot's current state and a chunk of $T{=}15$ actions produced by the policy, it predicts the resulting sequence of joint-position and gripper-position deltas, which are then integrated to obtain future absolute states.

Input representation. The model receives two groups of inputs:

- *State token*: the current 7-DOF joint position concatenated with the current scalar gripper position, forming a $(7{+}1)$-dimensional vector.
- *Action tokens*: a chunk of T joint-velocity commands ($T \times 7$) concatenated with T gripper actions ($T \times 1$), flattened to $T(7{+}1)$ dimensions.

Both groups are concatenated into a single input vector of dimension $(7{+}1)(T{+}1) = 128$ (for $T{=}15$).

Architecture. The adapter is a three-layer MLP with hidden size 512 and SiLU activations:

$$f_\theta : \mathbb{R}^{128} \longrightarrow \mathbb{R}^{T\times 8}.$$

The output is reshaped to $(T, 8)$ and split into predicted joint deltas ($T \times 7$) and predicted gripper deltas ($T \times 1$).

Normalization. All continuous inputs and targets are min-max normalized to $[-1, 1]$ using per-dimension 1st/99th-percentile bounds computed from the training set, which is more robust to outliers than global min/max. Gripper action commands are binarized ($\geq 0.5 \mapsto 1$, otherwise 0) prior to input, reflecting their discrete open/close semantics.

Loss function. The model is trained with a weighted MSE loss on the normalized delta targets:

$$\mathcal{L} \;=\; \mathcal{L}_{\text{joint}} \;+\; \lambda_g \, \mathcal{L}_{\text{gripper}}, \qquad \lambda_g = 5.0.$$

The gripper is up-weighted because it has a much smaller dynamic range than the joint dimensions and would otherwise be under-penalized relative to its importance in grasp and place predictions.

Inference. At test time the model predicts $(\hat{\Delta}_{\text{joint}},\ \hat{\Delta}_{\text{gripper}})$, denormalizes them, and integrates from the current state:

$$q_{t+k} = q_t + \hat{\Delta}_{\text{joint},k}, \qquad g_{t+k} = g_t + \hat{\Delta}_{\text{gripper},k}, \qquad k = 1, \ldots, T.$$

Training details. The adapter is trained for 15 epochs using Adam (lr=10^{-4}, batch size 128) on 50 hours of proprioceptive teleoperation data. Each training sample consists of a randomly drawn window of $T{+}1$ consecutive timesteps from an episode; the first timestep provides the current state and the remaining T timesteps provide the action chunk and delta targets.

A2 IMPLEMENTATION DETAILS

A2.1 ARCHITECTURE DETAILS

We use the VAE from Stable Diffusion 3 Esser et al. (2024) to encode 190×320 image frames from camera views into the latent space. Our efficient transformer architecture is a 32-layer transformer with 1536 hidden dimensions and 16 attention heads. Each layer comprises of a spatial layer that attends to all the patches in z_t and a causal temporal layer to attend over patches from prior observations. The actions and proproceptive states are normalized using statistics obtained with the training dataset of DROID. We obtain the reward annotations from Robometer Liang et al. (2026) and use the progress rewards to train the reward head and the critic. The reward and critic networks uses an AdaPool Brothers (2025) layer to compress the tokens to a vector, and has MLP layers with this vector concatenated with the CLIP embedding Radford et al. (2021) (provides a representation for the language instruction).

A2.2 TRAINING DETAILS

WEAVER Pretraining. WEAVER has 928M parameters in total and is trained for 1M gradient steps on 4×H100 GPUs for 10 days. The pretraining is done on the DROID dataset Team (2024). We use the same train-test split as provided in the codebase of Ctrl-World (Guo et al., 2026b). We also maintain an exponential moving average (EMA) of model weights during training with $\beta = 0.9999$. We use a learning rate warmup for the initial 10000 steps and keep a constant learning rate of $1e^{-4}$ post warmup. We provide the hyperparameters in Table 2.

WEAVER Finetuning. For complex tasks like pouring that is underrepresented in the pretraining dataset, the world model is inaccurate at predictions. To mitigate covariate shift and improve generations, a potential solution is to finetune the world model with task dataset. In this work, we finetune WEAVER on a small datasets of 250 trajectories (50 for each task) collected using the $\pi_{0.5}$ VLA. We finetune WEAVER for 16K gradient steps using a smaller learning rate of $2e^{-5}$. Other hyperparameters are similar to pretraining (as described in Table 2) and the training takes 6 hours on 4×H100 GPUs.

$\pi_{0.5}$ **Finetuning.** We follow the original setup of the droid finetuning configuration in openpi to finetune our policy. We use the open-sourced pre-trained $\pi_{0.5}-$droid checkpoint as the base policy. Normalization statistics are inherited from the original DROID checkpoint and held fixed throughout fine-tuning to preserve compatibility with the pretrained observation encoder. All runs with dataset size smaller than 5000 segments use a batch size of 32, a peak learning rate of 2.5×10^{-6} with cosine decay and no warmup, and language instructions sourced from task annotations. For datasets of size $\geq$5,000 trajectories, we fine-tune for 10,000 gradient steps with a cosine decay over 10,000 steps and warmup steps of 1000 and peak learning rate of 2.5e-5. For smaller datasets (1,000–2,000 trajectories), we reduce training to mitigate overfitting.

Reward Labeling. We label each trajectory with a per-frame progress reward using the Robometer evaluation server Liang et al. (2026). For each episode, we extract frames from the recorded video and downsampled to 1 fps using, selecting the right-camera view from the DROID setup. The sampled frames and the episode's language instruction are sent in a single forward pass to the Robometer eval server, which returns a per-frame *progress prediction* $\hat{r}_t \in [0, 1]$ representing the estimated fraction of task completion at frame t, along with an optional per-frame success probability $\hat{s}_t \in [0, 1]$.

Because frames are subsampled at 1 fps, the resulting reward sequence is shorter than the original video. We realign rewards to the full video length by linear interpolation: letting T_{orig} denote the original frame count and T_{sampled} the number of inferred frames, we place sampled values at positions $\{(T_{\text{orig}} - 1)\, i/(T_{\text{sampled}} - 1)\}_{i=0}^{T_{\text{sampled}}-1}$ and interpolate onto the integer grid $\{0, 1, \ldots, T_{\text{orig}} - 1\}$. We choose `reward_progress` (the interpolated progress signal) as our final reward annotation because it is more aigned with actual task outcome. We substract the reward progress by 1 to make the reward fall in $[-1, 0]$ as labels for training.

A2.3 INFERENCE

Inference noise schedules. We evaluate several deterministic schedules to map discrete inference index $i \in \{0, \ldots, K\}$ to the noise level $k \in [0, 1]$, where K is the number of denoising steps. We describe the different noise schedules compared– linear, sigmoid, power and cosine:

$$\textbf{Linear:} \quad k = \frac{i}{K},$$

$$\textbf{Sigmoid:} \quad k = \sigma\left(\alpha\left(\frac{i}{K} - 0.5\right)\right),$$

$$\textbf{Power:} \quad k = \left(\frac{i}{K}\right)^{0.5},$$

$$\textbf{Cosine:} \quad k = 1 - \cos\left(\frac{i\pi}{2K}\right),$$

where $\sigma(\cdot)$ is the logistic sigmoid and α controls the sharpness. For the sigmoid schedule, we normalize endpoints to be $k_0 = 0$ and $k_K = 1$. The linear schedule allocates steps uniformly, cosine

Name	Value
World Models	
Layers	32
Heads	16
Embedding dimension	1536
Head dimension	96
SPRINT probability	0.5
Reward Model and Critic	
MLP layers	2
Discount factor γ	.995
Return lambda λ	.95
Pretraining	
Batch size	32
Batch length	8
Memory frames (p)	6
Memory frame stride (m)	5
Optimizer	AdamW
Proprioceptive State loss scale ($\mathcal{L}_q$)	0.1
LR	$1e^{-4}$
Warmup steps	10000
EMA decay	.9999
Training Steps	1000000
Finetuning	
Batch size	32
LR	$2e^{-5}$
Warmup steps	200
EMA decay	.9999
Training Steps	16000

Table 2: **Hyperparameters.** We present the list of hyperparameters used for training WEAVER.

and power allocate more budget near low-noise regions, and sigmoid concentrates updates around the middle of the trajectory.

Rectified-Flow. To further reduce inference time and NFE for downstream tasks like test-time planning, we used ReFlow Liu et al. (2023) to post-train WEAVER-FT model, and call it WEAVER-ReFlow. The teacher and student model are initialised with a WEAVER-FT model where we freeze the teacher model. At each training iteration, we sample noise x^0 and predict future latents with the teacher model $\hat{x}^1$. This student model is updated with the predicted latent as the target using mean squared error loss given by: $\mathcal{L}^{\texttt{ReFlow}}(\phi) = \mathbb{E}_{x_t^0, \hat{x}_t^1, \tau}\left[\left\|(\hat{x}_t^1 - x_t^0) - f_\phi(\mathbf{z}_t^{\texttt{hist}}, \mathbf{z}_t^{\texttt{mem}}, \mathbf{a}_t, x_t^\tau, \tau)\right\|_2^2\right]$. The post-training with rectified flow is performed for 2K gradient steps on $4 \times H100$ (6 hours) with a learning rate of $2e^{-5}$.

A3 Additional World Model Evaluation Results

We provide additional results to evaluate WEAVER at coherent generations, impact of KV Cache on inference time, benefits of noise schedules, finetuning on task data, and post-training with rectified flow.

A3.1 World Model Evaluation

For a trajectory in validation dataset, we generated the rollout from the 20-th step, and compute the metrics using the generations for next 10s. We use the first 20 frames to initialize the memory and history for WEAVER and Ctrl-World. We report LPIPS Zhang et al. (2018), FID Heusel et al. (2017), and FVD Unterthiner et al. (2018) obtained using the ground truth videos. To obtain LPIPS, we utilize the functionality in `torchmetrics`[2] that uses the per-frame features obtained from `vgg` layers. To compute the FID, we use the implementation provided in `pytorch-fid`[3] repository. Our results on FVD are computed using the `Style-GAN-V`[4] repository. Here, we subsample multiple trajectories of 16 frames with a stride of 8 from each trajectory.

Apart from the quantitative metrics, we also qualitatively investigate the action controllability of the world model generation in Fig. 9. Starting from a shared wrist-camera observation at (T=0), we add 2cm along (+X), (-Y), or (-Z), per step to the end-effector pose and use inverse kinematics to compute the corresponding joint states while maintaining a fixed orientation.The computed joint states are used to condition the predicted rollout, with observations shown every two steps. For a commanded displacement of 16 cm, WEAVER predicts 15.5 cm along (+X) and 14.2 cm along both (-Y) and (-Z). These directionally consistent rollouts demonstrate the controllability of the generation from WEAVER.

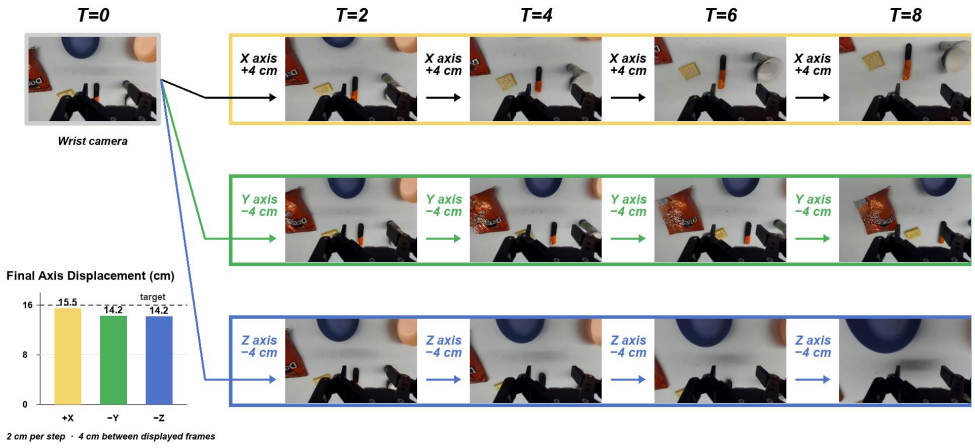


Figure 9: **Action controllability of WEAVER**. From the same wrist-camera observation at (T=0), the end effector is commanded to move 2 cm per step along (+X), (-Y), or (-Z), with orientation fixed. Frames are displayed every two steps (4 cm). After eight steps, WEAVER predicts displacements of 15.5, 14.2, and 14.2 cm, respectively, against a 16 cm target.

A3.2 Quantitative Results

Table 3 reports the comparison of WEAVER and Ctrl-World on DROID and OOD datasets across multiple metrics. We observe that performance of Ctrl-World deteriorates with lower NFE whereas WEAVER shows slight drop in performance with decrease in NFE. Moreover, with similar NFE values, our method is 3× faster at generating rollouts than Ctrl-World. In Fig. 10, we present the comparison of FID and inference time and observe that WEAVER with lower NFE of 8 outperforms Ctrl-World with large NFE of 50. We also include qualitative results of different NFEs with different world models in Fig. 14 and Fig. 15.

[2] https://github.com/Lightning-AI/torchmetrics
[3] https://github.com/mseitzer/pytorch-fid
[4] https://github.com/universome/stylegan-v

Method	NFE	*Exterior* LPIPS ↓	FID ↓	FVD ↓	*Wrist* LPIPS ↓	FID ↓	FVD ↓	Time (s) ↓
DROID								
Cosmos3-Nano	8	0.299	40.01	165.79	0.457	39.09	277.26	12.81
	16	0.298	34.13	137.20	0.460	32.11	190.07	16.30
	32	0.299	31.76	121.08	0.460	27.67	135.95	23.57
	50	0.302	32.19	129.75	0.462	26.27	115.28	32.46
Ctrl-World	8	0.169	31.63	116.14	0.407	52.40	347.69	8.14
	16	0.165	26.09	78.73	0.392	33.83	195.37	14.65
	32	0.168	23.63	63.55	0.389	27.14	114.87	27.67
	50	0.168	22.44	55.05	0.388	25.32	91.77	42.33
WEAVER	8	**0.117**	10.59	28.97	0.372	24.25	104.53	**2.53**
	16	**0.117**	10.20	27.83	**0.371**	21.50	90.72	4.78
	32	0.120	9.67	**25.94**	0.378	17.53	**63.36**	9.22
	50	0.122	**9.51**	26.54	0.378	**16.75**	66.89	14.25
OOD dataset								
Cosmos3-Nano	8	0.265	47.55	215.35	0.417	45.16	364.69	12.81
	16	0.263	42.44	182.39	0.420	42.23	295.09	16.30
	32	0.263	41.08	164.34	0.422	39.16	258.54	23.57
	50	0.265	41.54	151.55	0.423	38.42	234.84	32.46
Ctrl-World	8	0.193	48.90	226.29	0.374	51.26	434.84	8.14
	16	0.182	36.16	139.54	0.366	38.76	277.13	14.65
	32	0.183	32.18	105.38	0.365	33.73	173.15	27.67
	50	0.184	31.44	91.48	0.367	33.47	145.86	42.33
WEAVER	8	**0.154**	23.89	89.55	**0.364**	31.70	193.55	**2.53**
	16	0.155	23.95	88.27	**0.364**	30.77	184.62	4.78
	32	0.157	**23.45**	92.36	0.365	28.24	148.85	9.22
	50	0.159	23.48	**87.03**	0.371	**27.37**	**145.04**	14.25

Table 3: Comparison of WEAVER, Cosmos3-Nano and Ctrl-World at LPIPS, FID and FVD metrics. WEAVER generates with higher fidelity than Ctrl-World and has significantly better performance at low inference budgets. Here, NFE is Number of Function Evaluations and inference time is the time required to generate a 10s segment on a single H100 GPU.

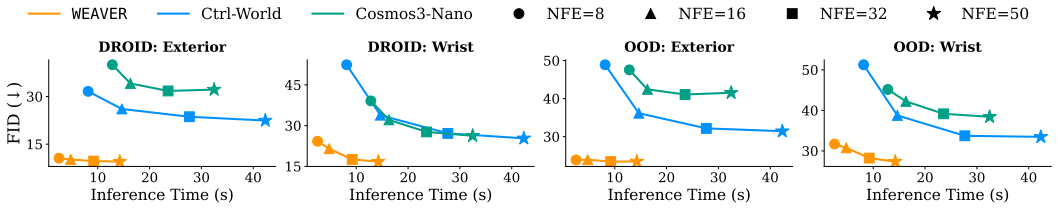


Figure 10: We compare FID vs inference time for WEAVER and Ctrl-World and find that WEAVER outperforms the baseline even when the baseline uses up to 16× more inference time

A3.3 Reducing inference time with KV Cache

During the iterative denoising process, the latents of memory and history frames are passed with a constant noise level $k = 1$. Since it does not change during this process, we compute the cache of key-value vectors of memory and history latents at the first denoising step. In Table 4, we report that KV Cache can reduce inference time by up to 30%.

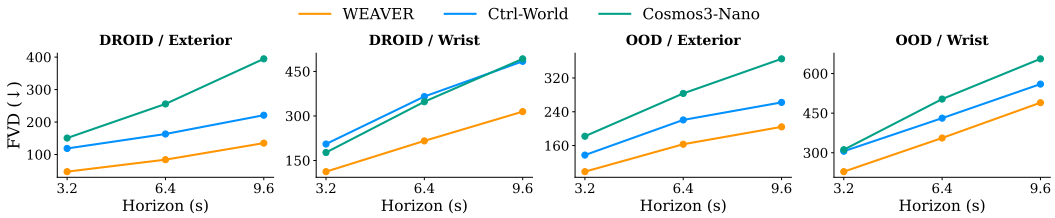


Figure 11: We report FVD at various horizon lengths (in seconds) and observe that WEAVER is consistently better at long-horizon rollouts compared to Ctrl-World and Cosmos3-Nano. Since computing FVD requires 16 frames to compute features, we divide the video into 3 segments (1-16 frames, 17-32 frames, 33-48 frames).

/ NFE	8	16	32	50
Without KV cache	3.30	6.54	12.98	20.21
With KV cache	2.53	4.78	9.22	14.25

Table 4: We report the inference time (in seconds) taken to generate a 10s trajectory on a single H100 GPU at different NFE, and observe that using KV Cache can reduce inference time by upto 30%.

A3.4 Noise schedules during inference

Table 5 compares different noise schedules where we observe that both power and cosine noise schedules perform better than sigmoid and linear noise schedules. Since the world model needs to generate with higher fidelity, the noise schedules that allocate more bandwidth at low noise regions aids in generating fine-grained details.

A3.5 Finetuning

In Table 6, we observe that finetuning significantly improves performance and with NFE=16 it performs better than `WEAVER` with NFE=50. To provide a fair comparison with Ctrl-World, we finetune the baseline for 20K gradient steps on $4 \times H100$ and see that the finetuned Ctrl-World (called Ctrl-World-FT) performs better than the pretrained model. However, `WEAVER`-FT outperforms Ctrl-World-FT across metrics and the performance is still larger with low NFE=16. This further demonstrates that finetuning does not help in reducing inference time for Ctrl-World. We also provide qualitative results of the rollouts generated from Ctrl-World, `WEAVER`, `WEAVER-FT` in Fig. 16, Fig. 17 and Fig. 18.

		Exterior			*Wrist*		
Method	**Schedule**	**LPIPS ↓**	**FID ↓**	**FVD ↓**	**LPIPS ↓**	**FID ↓**	**FVD ↓**
				DROID			
Ctrl-World	linear	0.165	26.09	78.73	0.392	33.83	195.37
`WEAVER`	linear	**0.117**	11.32	**26.38**	0.375	24.43	98.82
	sigmoid	**0.117**	10.88	29.76	0.375	22.69	104.89
	power	**0.117**	10.57	27.93	**0.369**	**21.17**	91.51
	cosine	**0.117**	**10.20**	27.83	0.371	21.50	**90.72**
				New dataset			
Ctrl-World	linear	0.182	36.16	139.54	0.366	38.76	277.13
`WEAVER`	linear	0.157	25.37	96.16	0.367	33.24	217.65
	sigmoid	0.156	24.83	93.15	0.367	32.48	216.89
	power	**0.155**	**23.82**	**84.91**	**0.363**	31.60	185.30
	cosine	**0.155**	23.95	88.27	0.364	**30.77**	**184.62**

Table 5: We see that cosine and power noise schedules allocates higher budget at low noise scales and perform better than linear and sigmoid schedules. The numbers are reported with NFE=16.

Method	NFE	*Exterior* LPIPS ↓	FID ↓	FVD ↓	*Wrist* LPIPS ↓	FID ↓	FVD ↓
Task Data (OOD)							
Ctrl-World	50	0.184	31.44	91.48	0.367	33.47	145.86
Ctrl-World-FT	16	0.140	29.89	81.16	0.295	34.77	283.00
	50	0.142	25.96	58.27	0.292	25.80	134.69
WEAVER	50	0.159	23.48	87.03	0.371	27.37	145.04
WEAVER-FT	4	0.116	16.62	50.68	0.304	33.32	219.15
	16	0.118	14.24	40.50	0.303	23.39	146.35
	50	0.121	13.69	40.09	0.308	18.73	99.81
WEAVER-REFLOW	4	0.123	14.95	44.30	0.312	23.56	138.67

Table 6: Compare the finetuned variants of WEAVER and Ctrl-World on Task dataset (OOD) where WEAVER-FT outperforms the baselines. Moreover, the post-training step (WEAVER-REFLOW) further helps to reduce inference budget.

Table 7: Inference time breakdown for test-time planning on A6000 Ada GPU. We report the runtime of each component with different horizons of world model imaginations across 10 function calls.

Component	Notation	Batch Size	Horizon	Runtime (s) ↓
Policy sampling	$\pi_\theta(a_{t:t+h} \mid o_t, \ell)$		–	$0.1979_{\pm 0.0002}$
Dynamics model (WEAVER)	$f_\phi(\hat{z}_{t+1:t+h} \mid \mathbf{z}_t^{\text{mem}}, \mathbf{z}_t^{\text{hist}}, a_{t:t+h})$	4	9	$1.0203_{\pm 0.0100}$
		4	12	$1.2493_{\pm 0.0170}$
		4	15	$1.4547_{\pm 0.0160}$
		1	15	$0.4476_{\pm 0.0049}$
Dynamics Model (Ctrl-World)	$f_\phi(\hat{z}_{t+1:t+h} \mid \mathbf{z}_t^{\text{mem}}, \mathbf{z}_t^{\text{hist}}, a_{t:t+h})$	4	15	$29.4244_{\pm 1.0162}$
		1	15	$7.4236_{\pm 0.1201}$
Reward inference	$R(\hat{z}_{t+1:t+h}, \ell)$	4	–	$0.0006_{\pm 0.0002}$
Critic inference	$V(\hat{z}_{t+h}, \ell)$	4	–	$0.0005_{\pm 0.0001}$

A3.6 Posttraining with Rectified Flow

In table 6, we present the results of WEAVER-ReFlow with small inference budget and observe that it reduces the performance gap with WEAVER-FT evaluated with a large NFE=16. This makes it suitable for test-time steering as observed in Section 5.2.3.

A4 Experiments on BridgeV2 dataset

To demonstrate the efficacy of WEAVER on an additional benchmark, we conduct experiments on the BridgeData V2 dataset (Walke et al., 2023). BridgeData V2 contains 60,096 robot trajectories spanning 24 environments and 13 manipulation skills. The data were collected using a 6-DoF WidowX 250 robot arm. In our experiments, we use only the fixed, over-the-shoulder RGB camera, resulting in a single-view prediction setting. We adopt the training and evaluation splits released with the IRASim (Zhu et al., 2025) codebase.

We train WEAVER for 265K gradient steps with a per-GPU batch size of 8. Training takes approximately 12 hours on 8 × NVIDIA GB200 GPUs. For this dataset, we use a 525M-parameter configuration of WEAVER, with an embedding dimension of 1152 and 18 attention heads (other hyperparameters are similar to Table 2).

Baselines We compare against WorldGym (Quevedo et al., 2026), an autoregressive action-conditioned video world. We also compare against IRASim (Zhu et al., 2025), a fine-grained action-conditioned video world model that predicts future robot–object interactions from historical observations and action trajectories. Finally, we train two variants of WEAVER using the same data split and compute budget, enabling a controlled comparison of the architectural and training choices.

Table 8: Results on BridgeV2 dataset where we observe that WEAVER outperforms WorldGym and IRASim on all metrics.

Model	FID ↓	FVD ↓	LPIPS ↓
IRASim	33.66	203.66	0.3237
WorldGym	10.56	68.08	0.2157
WEAVER	**9.57**	**19.08**	**0.1845**

Table 9: Ablations on the choice of visual encoder and proprioceptive state prediction.

Model	FID ↓	FVD ↓	LPIPS ↓
no-proprio	10.93	22.11	0.1864
SVD Encoder	22.24	59.24	0.2059
WEAVER	**9.57**	**19.08**	**0.1845**

A4.1 Results

Table 8 presents the comparison of IRASim (Zhu et al., 2025), WorldGym (Quevedo et al., 2026) and WEAVER. We observe that our method outperforms baselines across FIC, FVD and LPIPS. Remarkably, WEAVER achieves around 3.5× lower FVD compared to WorldGym which is another diffusion based world model and uses SD3 encoder.

Choice of image encoder. We compare the SD3 and SVD encoders and find that WEAVER performs substantially better with the SD3 encoder, as shown in Table 9. Notably, even when using the SVD encoder, WEAVER continues to outperform WorldGym on FVD and LPIPS (Table 8). With the SD3 encoder, WEAVER further improves across all metrics, with particularly large gains in FVD. These results suggest that, beyond the choice of visual encoder, WEAVER benefits from its world-model architecture, training procedure, and inference design.

Proprioceptive state prediction. We additionally train a variant of WEAVER that does not explicitly predict proprioceptive states, denoted as WEAVER (no-proprio). Proprioceptive states are still provided as conditioning inputs during training. At inference time, we evaluate two variants: one using ground-truth proprioceptive states and another estimating the states from the executed actions. As shown in Table 9, explicitly predicting proprioceptive states improves performance across metrics.

A5 Additional Downstream Application Results

A5.1 Policy Evaluation Results

We provide the full policy evaluation rollouts in Fig. 12. Both Ctrl-World and WEAVER struggle to accurately predict policy performance, especially on challenging tasks involving dynamic manipulation, such as pouring beans, and deformable object manipulation, such as bag and towel manipulation.

For the **PnP Bag** task, grasping the bag is particularly challenging because the world model must accurately infer the gripper depth across two camera views while also modeling the contact dynamics

Table 10: Comparison of policy evaluation correlation across different world models. We report RMSE, Spearman correlation, Pearson correlation, and MMRV.

Method	**RMSE ↓**	**Spearman ↑**	**Pearson ↑**	**MMRV ↓**
Ctrl World	0.517	0.410	0.459	0.278
Ours	0.473	0.517	0.468	0.240
Ctrl World-FT	0.399	0.677	0.673	0.200
Ours-FT	**0.174**	**0.817**	**0.846**	**0.104**



Figure 12: **Policy Evaluation Results**. We show policy evaluation results for all five tasks across four world models.

between the gripper and the deformable object. These challenges become more pronounced as the prediction horizon increases. In contrast, `WEAVER-FT` substantially improves evaluation accuracy through finetuning as shown in Fig. 19 and Fig. 20. Future work could further improve long-horizon prediction by designing better memory and history representations, enabling the model to better reason about occlusions and deformable object dynamics.

For **Pour Beans** task, predicting the dynamics of beans is also challenging for the existing world models. In Fig. 12, only `WEAVER-FT` can successfully predict the scattered beans on the table while all other methods fail to predict any beans on the table.

For quantitative metrics, we evaluate different world models on five different policies with 5 different tasks, resulting in 25 data points per method. In addition to Pearson correlation and MMRV, we also report RMSE and Spearman rank correlation (Wissler, 1905). Across these metrics, we observe a consistent trend: `WEAVER-FT` achieves the strongest correlation and lowest prediction error. In addition, Ctrl-World-FT outperforms `WEAVER` with additional finetuning data but `WEAVER` consistently outperforms Ctrl-World in zero-shot policy evaluation on out-of-distribution task dataset across all metrics. The full quantitative results are shown in Table 10.

A5.2 Reward Model Accuracy and Data Filtering

We use RoboMeter Liang et al. (2026) as an off-the-shelf reward model to provide supervision for training our reward prediction head. Since the quality of this supervision depends on the reliability of RoboMeter on our task distribution, we first evaluate its agreement with ground-truth task outcomes. As shown in Table 11, using a threshold of $\tau = 0.83$ calibrated on the training set, RoboMeter achieves an overall binary success-classification accuracy of **83%** across five manipulation tasks. Performance is particularly strong on bag, pour, and towel, each reaching 90% accuracy. Most errors are false negatives (12/17 errors), suggesting that RoboMeter is generally conservative: it more often assigns a low score to a successful rollout than incorrectly labels a failed rollout as successful. This

is useful for us to use that for data filtering because introducing failure data would bring additional noise in finetuning. In particular, there are only five false positives across 100 evaluated trajectories.

Binary classification, however, only evaluates whether the reward can separate successful from unsuccessful trajectories at a single operating threshold. We therefore additionally evaluate whether the continuous RoboMeter score preserves relative trajectory quality. Table 12 reports pairwise ranking accuracy, where RoboMeter is asked to assign a higher reward to the better trajectory according to ground-truth labels. Across 456 trajectory pairs, RoboMeter correctly ranks **87.3%** of the pairs, with particularly high agreement on bag (97.9%), pour (94.0%), and towel (90.6%). While performance is weaker on marker, the overall ranking accuracy indicates that the reward signal preserves useful relative quality information beyond binary success detection.

Together, these results suggest that RoboMeter provides a sufficiently reliable off-the-shelf supervisory signal for our tasks. We therefore use RoboMeter-generated rewards as labels for training our lightweight reward head, rather than requiring additional manually annotated reward data. Importantly, RoboMeter is used only to provide training supervision: at deployment, our reward head directly predicts the reward from the latent states, avoiding the need and additional latency to invoke RoboMeter online.

Table 11: Reward model accuracy for binary success prediction. We use the decision threshold $\tau = 0.83$, calibrated on the training set, and report accuracy on held-out rollouts.

Task	Accuracy	TP	TN	FP	FN
Bag	90% (18/20)	11	7	1	1
Pour	90% (18/20)	4	14	0	2
Towel	90% (18/20)	7	11	1	1
Stack	70% (14/20)	6	8	0	6
Marker	75% (15/20)	4	11	3	2
Overall	**83% (83/100)**	**32**	**51**	**5**	**12**

Table 12: Pairwise ranking accuracy of the RoboMeter reward model against ground-truth trajectory preferences. Brackets denote 95% confidence intervals.

Task	# Pairs	Pairwise Accuracy
Pour	84	94.0% [86.8, 97.4]
Stack	96	87.5% [79.4, 92.7]
Bag	96	97.9% [92.7, 99.4]
Marker	84	64.3% [53.6, 73.7]
Towel	96	90.6% [83.1, 95.0]
Overall	**456**	**87.3% [83.9, 90.0]**

We next investigate the accuray of the predicted reward when we couple the world model prediction with the reward prediction. We first construct an oracle where we use Robometer over Ground-Truth (GT) observations. Our method generates reward from our model's reward head over latent states of future steps. Robometer (Ctrl-World) instead generates the images of future steps and Robometer predicts over that. In Table 13, `WEAVER` achieves **74.7%** pairwise ranking accuracy, substantially outperforming the Ctrl-World reward at **62.3%**. Our model improves over Ctrl-World on every evaluated task, with particularly large gains on pour (79.8% vs. 57.1%) and towel (73.2% vs. 60.4%). The remaining gap to RoboMeter (GT) (87.3%) is expected, since RoboMeter (GT) serves as an oracle reward to supervise our reward training. Ctrl-World generates worse out-of-distribution predictions that confuses the RoboMeter reward model, which contributes to the worse performance across all tasks.

A5.3 Policy Improvement Results

We provide additional qualitative results of policy improvement in Fig. 13. These examples show that the base policy often suffers from imprecise grasping and placement, as well as insufficient

Table 13: Pairwise reward ranking accuracy on held-out trajectories. For each task, we form all success–failure pairs from 20 trajectories and measure whether each reward model assigns a higher score to the successful trajectory. RoboMeter is evaluated directly against ground-truth outcomes, while **Ours** uses our model's predicted reward head. RoboMeter(Ctrl-World) uses Ctrl-World to predict future images and run Robometer to predict the reward over those predicted images. Brackets denote 95% confidence intervals.

Task	# Pairs	RoboMeter (GT)	Ours	RoboMeter (Ctrl-World)
Pour	84	94.0% [86.8, 97.4]	79.8% [70.0, 87.0]	57.1% [46.5, 67.2]
Stack	96	87.5% [79.4, 92.7]	76.0% [66.6, 83.5]	66.7% [56.8, 75.3]
Bag	96	97.9% [92.7, 99.4]	80.5% [71.5, 87.2]	72.9% [63.3, 80.8]
Marker	84	64.3% [53.6, 73.7]	63.0% [52.3, 72.5]	52.3% [41.8, 62.7]
Towel	96	90.6% [83.1, 95.0]	73.2% [63.6, 81.0]	60.4% [50.4, 69.6]
Overall	**456**	**87.3% [83.9, 90.0]**	**74.7% [70.5, 78.5]**	**62.3% [57.7, 66.6]**

adjustment during dynamic manipulation. We also observe that the base policy tends to produce larger per-step motions, resulting in unstable robot control. In contrast, the finetuned policy substantially reduces these large movements and sharpens the action distribution, leading to smoother and more stable execution.

We also note that the RoboMeter reward labels are not perfect. For the **PnP Marker** task, we observe cases where the reward model fails to distinguish fine-grained placement accuracy, which can introduce noise into the predicted rewards. Future work could improve reward supervision by collecting more diverse failure data to train a more general and precise reward model. To mitigate the effect of noisy reward labels, we set the advantage threshold to 0.1, which helps prevent low-quality segments from being selected for finetuning and potentially degrading policy performance. As shown in Fig. 4, our filtering procedure is able to select the best action samples among the candidates.

A5.4 Test-Time Planning Results

Inference-Time Latency. Table 7 reports the inference-time breakdown of test-time planning on an A6000 Ada GPU. Overall, the runtime is dominated by the dynamics model imagination, while reward and critic inference are negligible, taking less than 0.001s each. For WEAVER, the dynamics runtime increases moderately with the imagination horizon: from 1.0203s at horizon 9, to 1.2493s at horizon 12, and 1.4547s at horizon 15 with batch size 4. Including policy sampling, reward inference, and critic inference, the full planning latency is approximately 1.22s, 1.45s, and 1.65s for horizons 9, 12, and 15, respectively. Therefore, our world model can achieve 33.5 FPS for future generation and obtain 10.5 Hz control frequency for test-time planning with batch size of 4.

Compared to Ctrl-World, WEAVER substantially reduces latency during imagination. At horizon 15 and batch size 4, WEAVER takes 1.4547s for dynamics prediction, while Ctrl-World requires 29.4244s, corresponding to a $20.2\times$ speedup. The same trend holds at batch size 1, where WEAVER takes 0.4476s compared to 7.4236s for Ctrl-World, yielding a $16.6\times$ speedup. These results show that WEAVER enables substantially lower-latency test-time planning, making repeated world-model imagination practical during policy execution.

A6 Limitations

While WEAVER demonstrates the promise of large-scale world models for policy evaluation, policy improvement, and test-time planning, several limitations remain.

A6.1 Partial Observability

Our world model relies primarily on visual observations, which provide only partial access to the underlying physical state. During manipulation, task-relevant information such as object contacts, grasp stability, applied forces, or occluded object geometry may be hidden from all available camera views. This limitation is especially pronounced for wrist-camera observations, where the viewpoint

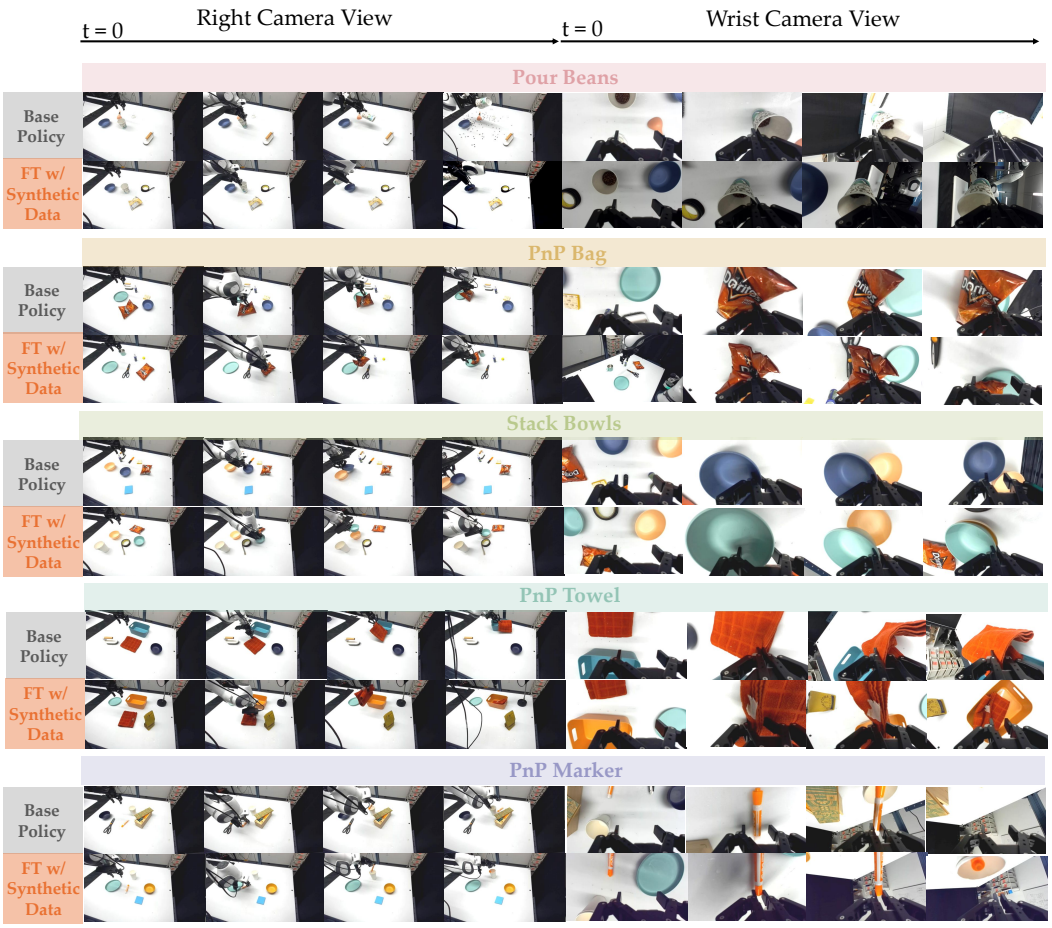


Figure 13: **Policy Improvement Results**. We demonstrate the rollouts for five tasks among the base policy and policy FT w/ Synthetic Data. With WEAVER generated synthetic data, policy finetuning can improvement upon all tasks.

changes continuously, and for cluttered scenes where objects may leave the field of view or become occluded by the gripper. Although memory and multi-view conditioning mitigate this issue, purely visual prediction may still fail when the missing state cannot be inferred from image history alone. Incorporating additional sensing modalities, such as tactile feedback, force-torque sensing, or depth, may improve state estimation and long-horizon prediction under occlusion.

A6.2 Complex Deformable and Dynamic Interactions

Deformable-object manipulation and dynamic manipulation remain challenging for learned world models. Objects such as towels, bags, and granular materials exhibit high-dimensional, history-dependent dynamics that are difficult to capture from limited robot data. Small errors in predicted contact, grasp location, or object configuration can compound over time and lead to qualitatively incorrect rollouts. This is particularly evident in tasks such as pouring, where the motion of granular material depends sensitively on cup pose, velocity, and contact with the container. Future work may improve prediction fidelity by incorporating physics priors, hybrid neural-physics models, or neural simulators specialized for deformable and granular dynamics.

A6.3 Limited Planning Horizon at Test Time

Although our inference acceleration strategies make test-time planning feasible with a large generative world model, latency still limits online planning to a single action chunk. As a result, the planner can improve near-term action selection but cannot yet perform long-horizon lookahead. This restricts its ability to reason about delayed consequences or multi-stage recovery behaviors. Further improvements in sampling efficiency, model distillation, value estimation, or hierarchical planning could enable longer-horizon online reasoning while maintaining real-time control.

A6.4 Data Coverage and Embodiment Diversity

Our world model is pretrained primarily on DROID, which provides large-scale robot interaction data but is still tied to a specific robot embodiment and data collection setup. This may limit generalization to substantially different robots, camera configurations, end-effectors. In addition, some task dynamics in our evaluation, such as granular pouring, are underrepresented in the pretraining data. Scaling world-model training to more diverse sources, including cross-embodiment robot datasets, simulation data, and human videos, may improve robustness and broaden the range of behaviors that can be accurately imagined.

A6.5 Noisy Reward Supervision

Our latent reward and critic heads are trained using labels from an off-the-shelf reward model. While this enables efficient latent-space evaluation, the resulting supervision can be noisy or incomplete, especially for subtle failure modes. For example, a reward model may fail to distinguish between visually similar but semantically different outcomes, or may be insensitive to small errors in contact, placement, or task completion. Such noise can affect both policy evaluation and downstream policy improvement. A more reliable reward model trained on large-scale robot success and failure data, potentially with calibrated uncertainty, would likely improve the reliability of imagined rollout evaluation.

Overall, these limitations suggest that future progress will require not only larger and faster world models, but also richer sensing, broader data coverage, stronger physical inductive biases, and more accurate reward supervision.

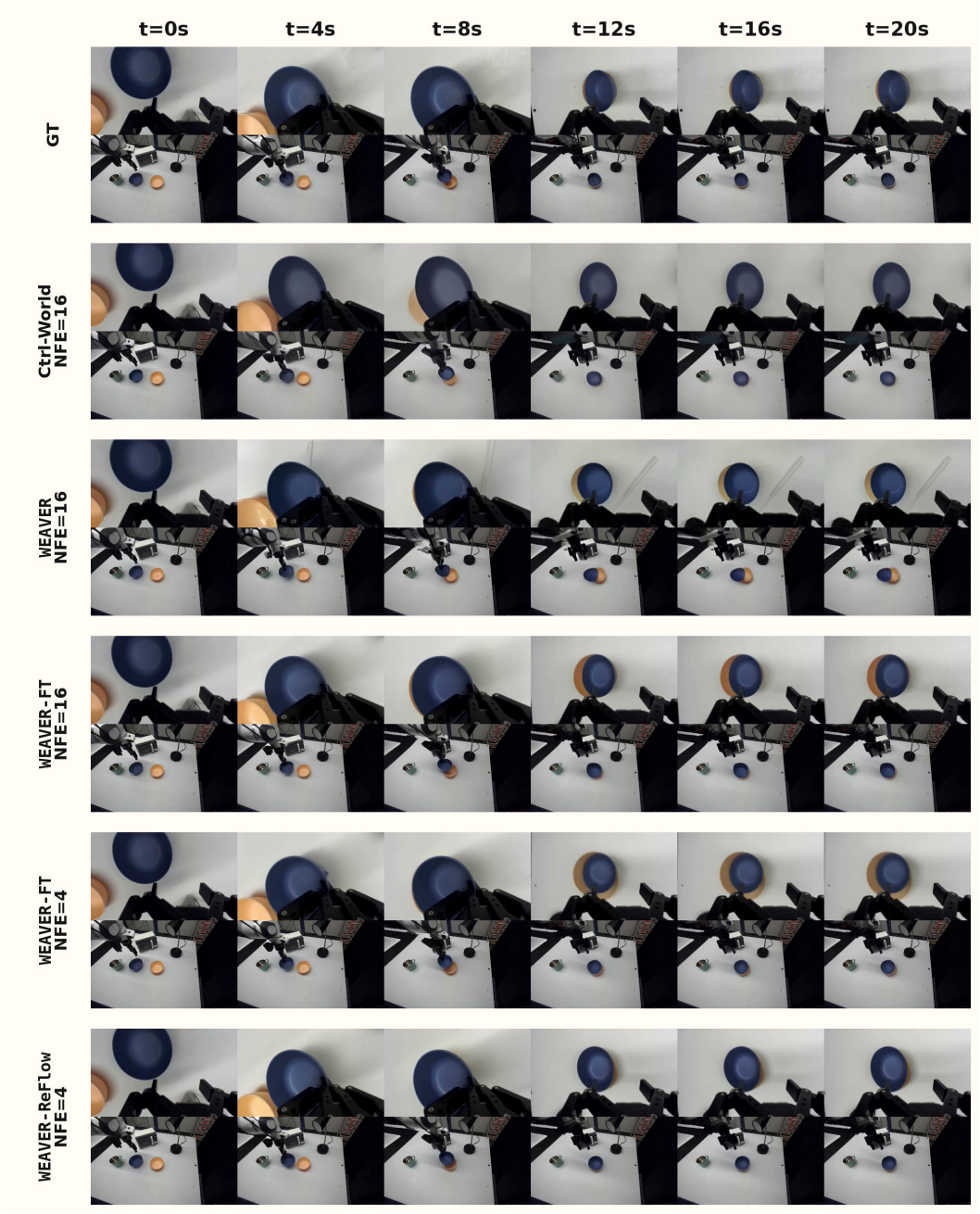


Figure 14: We compare the rollouts on task obtained from Ctrl-World, WEAVER, WEAVER-FT and WEAVER-REFLOW at different NFE values of 4 and 16. We generate rollouts for 20 seconds and present predicted camera views at every 4 second. We observe that WEAVER-REFLOW is better than WEAVER-FT at NFE=4 and has comparable performance with other models using NFE=16.

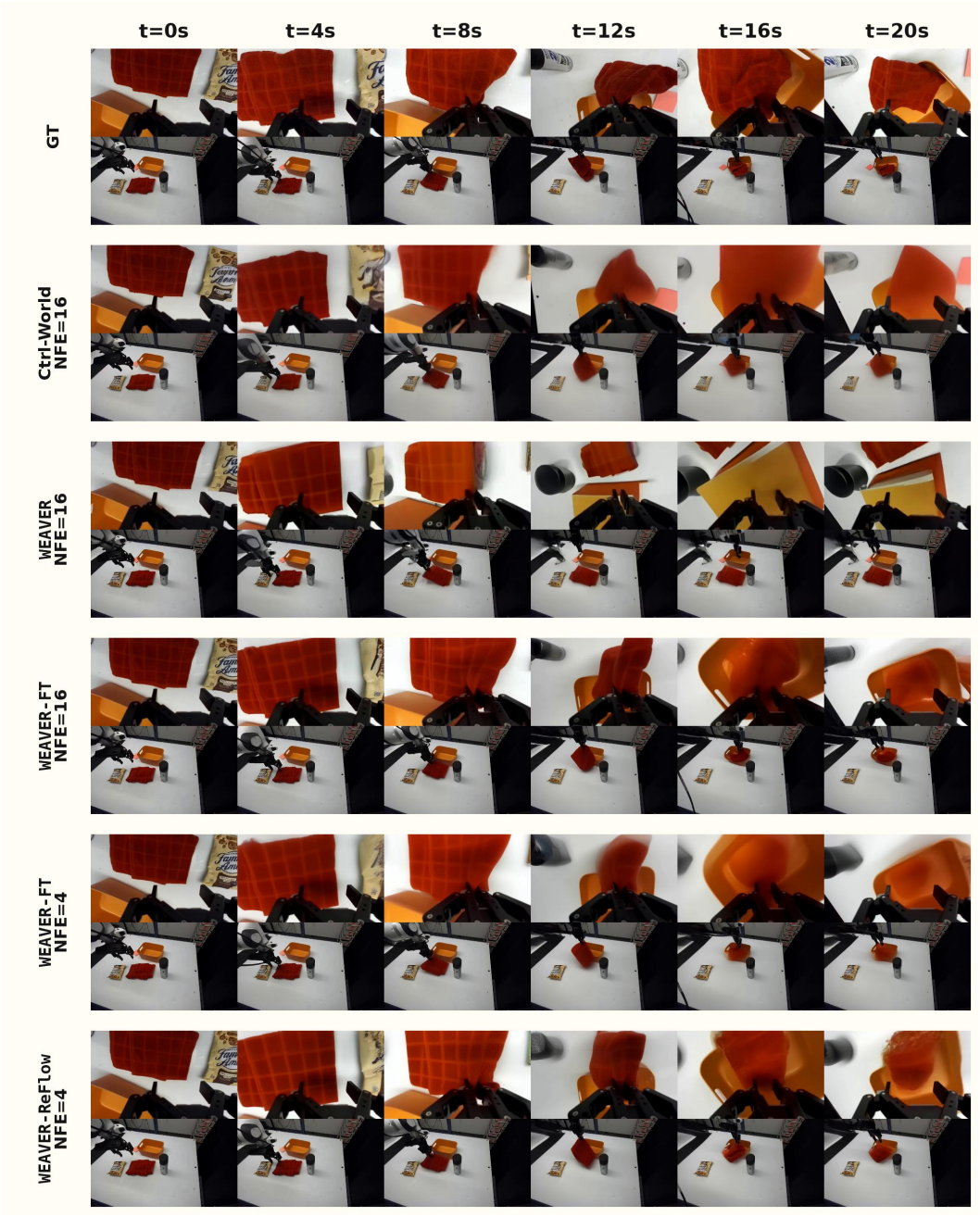


Figure 15: We compare the rollouts on task obtained from Ctrl-World, WEAVER, WEAVER-FT and WEAVER-REFLOW at different NFE values of 4 and 16. We generate rollouts for 20 seconds and present predicted camera views at every 4 second. We observe that WEAVER-REFLOW is better than WEAVER-FT at NFE=4 and is more consistent than Ctrl-World and WEAVER with NFE=16.

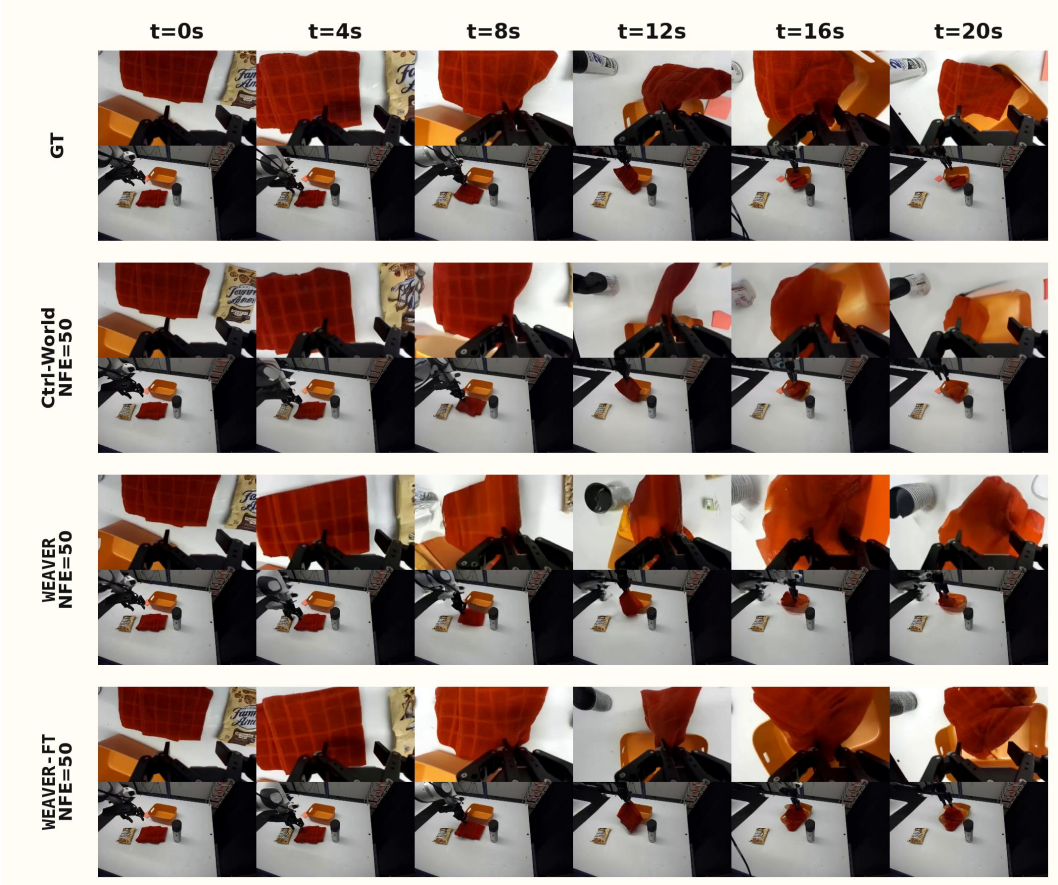


Figure 16: We compare the rollouts on task obtained from Ctrl-World, WEAVER and WEAVER-FT and WEAVER-REFLOW at different NFE=50. We generate rollouts for 20 seconds and present predicted camera views at every 4 second. We observe that Ctrl-World struggles at retraining information about the towel after 12 seconds and WEAVER-FT is more consistent with the ground truth.

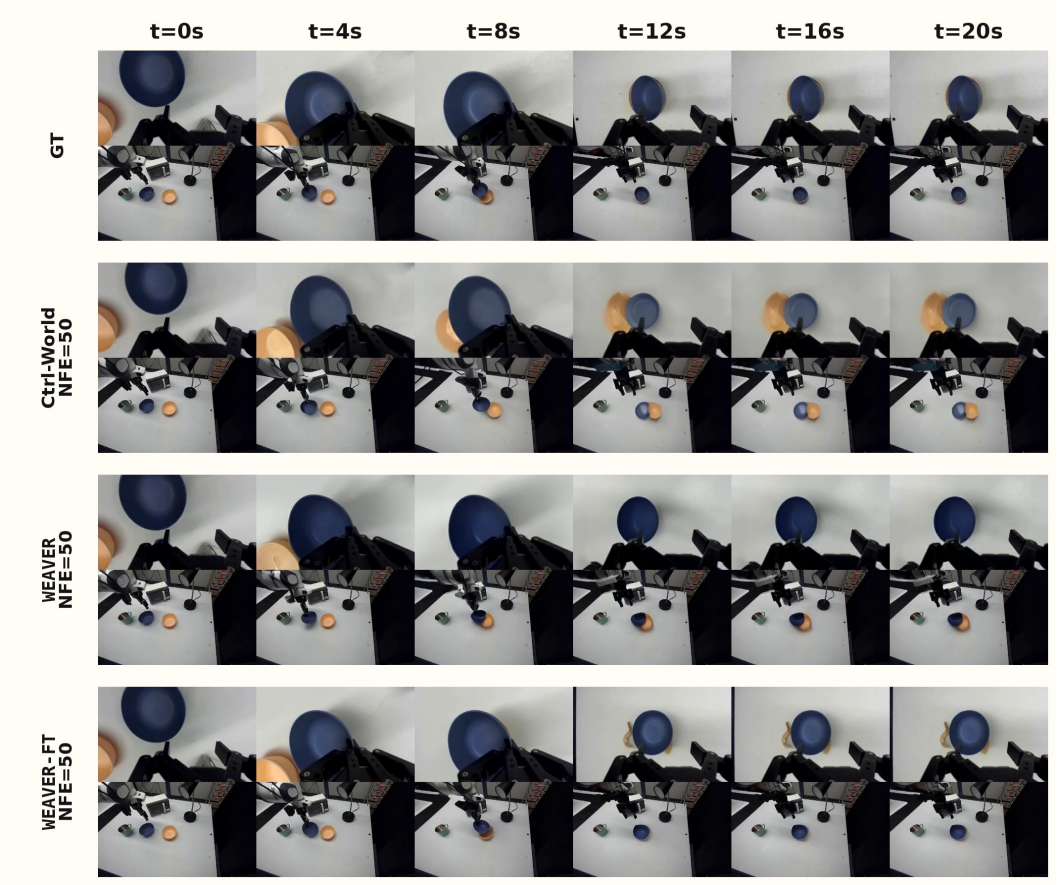


Figure 17: We compare the rollouts on task obtained from Ctrl-World, WEAVER and WEAVER-FT and WEAVER-REFLOW at different NFE=50. We generate rollouts for 20 seconds and present predicted camera views at every 4 second. We observe that WEAVER and WEAVER-FT is better than Ctrl-World at predictions.

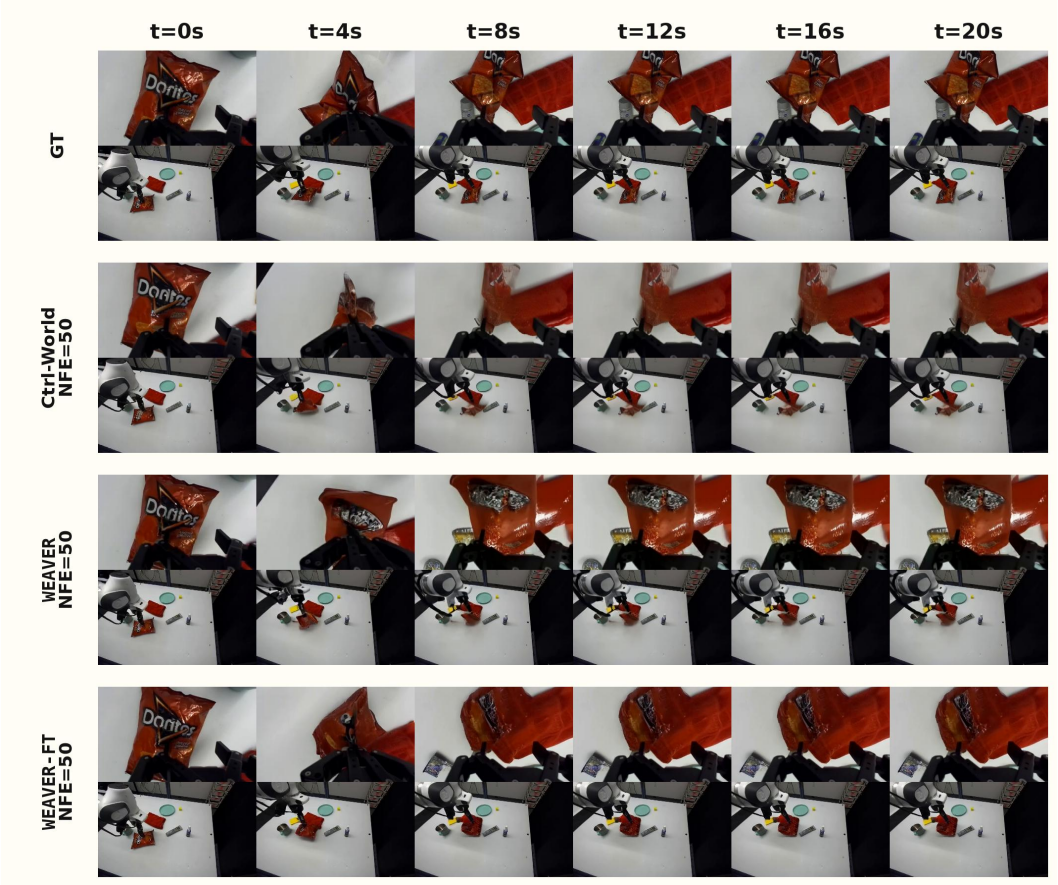


Figure 18: We compare the rollouts on task obtained from Ctrl-World, WEAVER and WEAVER-FT and WEAVER-REFLOW at different NFE=50. We generate rollouts for 20 seconds and present predicted camera views at every 4 second. We see that Ctrl-World struggles to predict the object from t=4s compared to WEAVER and WEAVER-FT.

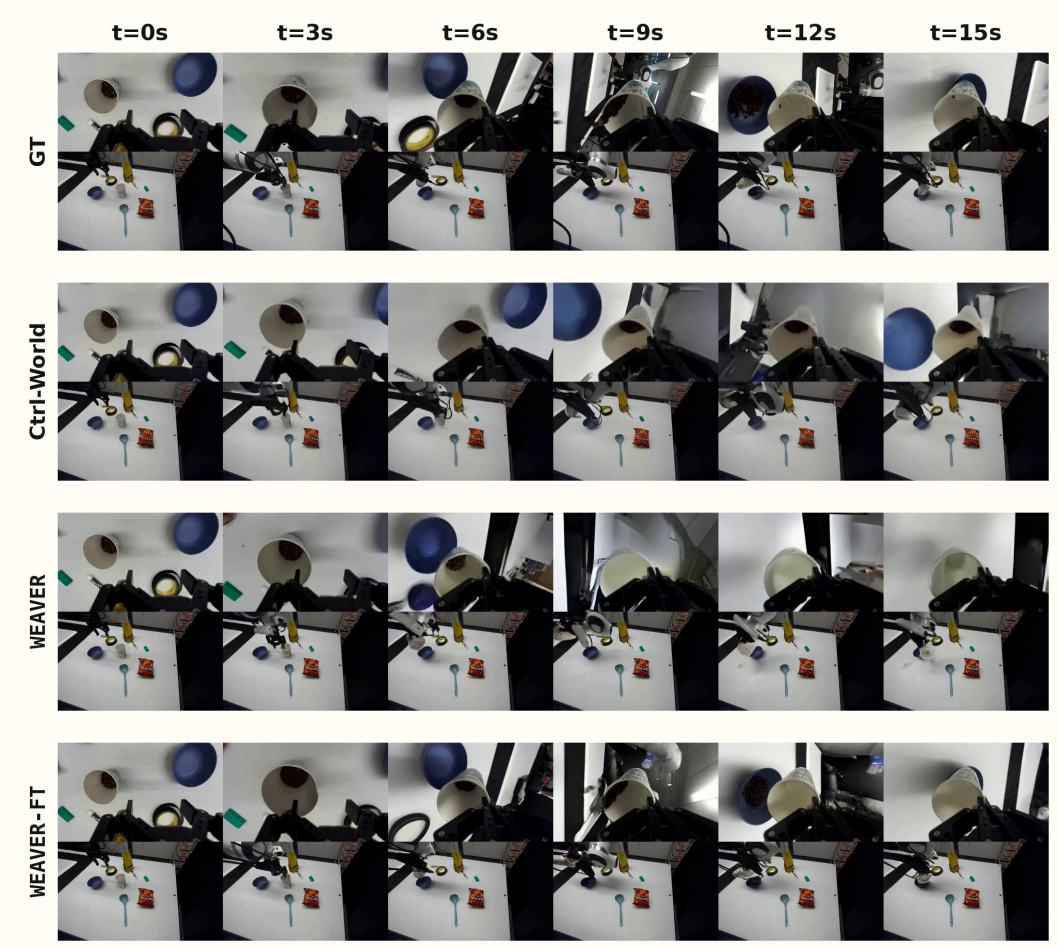


Figure 19: We compare the rollouts on Pour Beans task obtained from Ctrl-World, WEAVER and WEAVER-FT for policy evaluation at NFE=50. We generate rollouts for 15 seconds and present predicted camera views at every 3 second. We see that Ctrl-World and WEAVER struggles to predict the beans in the bowl at t=12s compared to WEAVER-FT.

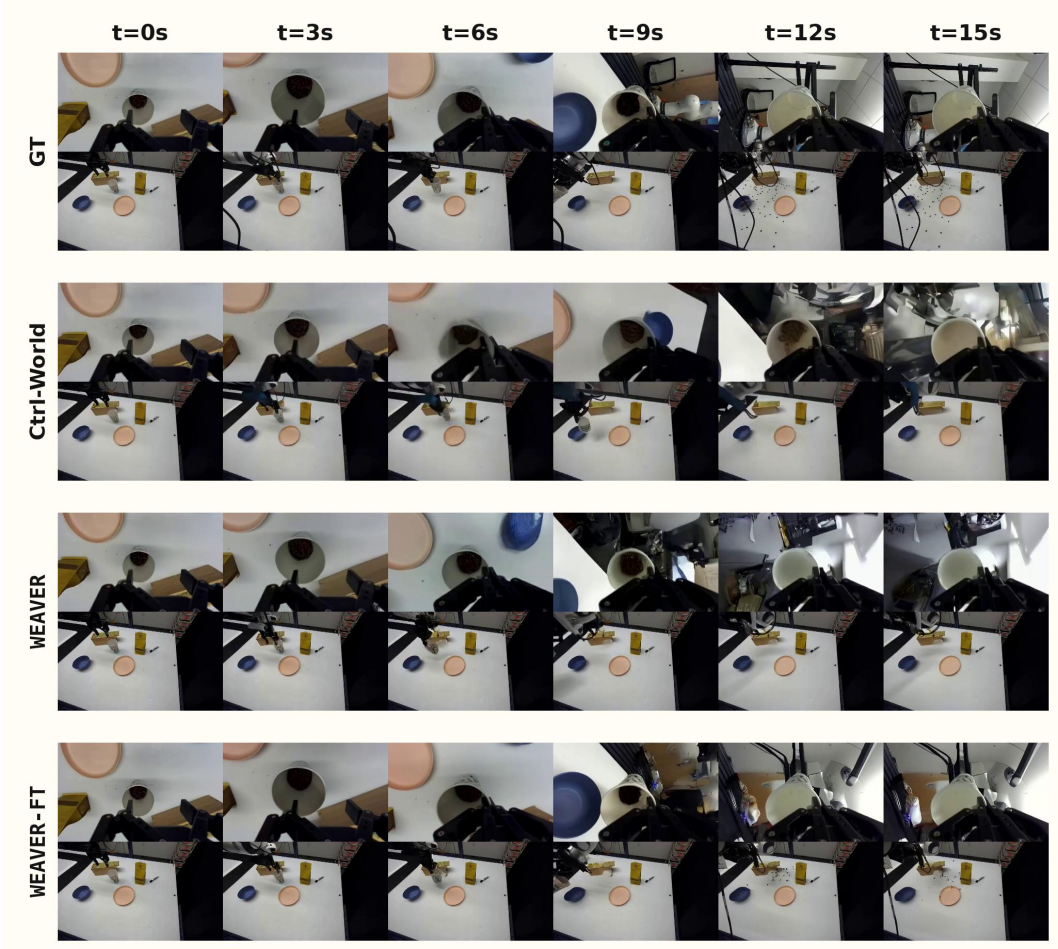


Figure 20: We compare the rollouts on Pour Beans task obtained from Ctrl-World, `WEAVER` and `WEAVER-FT` for policy evaluation at NFE=50. We generate rollouts for 15 seconds and present predicted camera views at every 3 second. We see that Ctrl-World and `WEAVER` struggles to predict the beans on the table at t=15s compared to `WEAVER-FT`.